Convex Optimization

through the lens of algorithms and applications

Contents

Part One

1. Background knowledge

This section gives an overview of the basic concepts required to understand the material of the upcoming chapters. The summary captures four topics: convex sets and convex functions, polytopes and polyhedra, cones, and linear programming. This section's material does not fully cover all of the mentioned topics but instead serves as a gentle repetition. If the presented material is new to the reader, additional reading on these topics is recommended. For this reason, in this section, very few proofs are provided. However, missing proofs can be found in the references where a deeper understanding of these topics can be acquired.

1.1 Convex sets and convex functions

In this section, we discuss the main concepts of convex sets and convex functions. We start with the definition of a line segment.

Definition 1.1.1 — Line segment. Let $p \neq q$ be two points in $\mathbb{R}^n$. The set

$$\{\lambda p + (1-\lambda)q \mid 0 \leq \lambda \leq 1\}$$

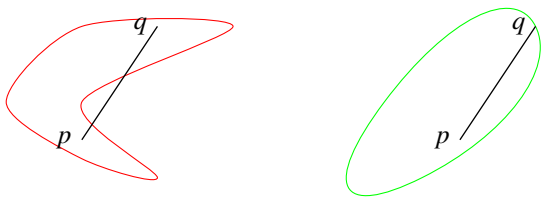
实数集

is the *line segment* between the points p and q, which we denote by $[p,q]$.

We are now ready to define a convex set.

Definition 1.1.2 — Convex set. A set $S \subseteq \mathbb{R}^n$ is *convex* if $[p,q] \subseteq S$ for all $p,\ q \in S$.

■ **Example 1.1.3** An example of a non-convex set on the left and a convex set on the right.



线段的集合，线段上所有的点都属于集合S

■

Convex sets admit many interesting properties, some of them are listed below.

Proposition 1.1.4 Let $S, T \subseteq \mathbb{R}^n$ be convex sets. Then

1. $S \cap T$ is convex,
2. the image of S under affine transformation is convex. More precisely for $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ the set,

$$\{A \cdot x + b \mid x \in S\}$$

is convex,

3. the *Minkowski sum* of sets S and T, i.e.

$$S \oplus T := \{x + y \in \mathbb{R}^n \mid x \in S,\ y \in T\}$$

is convex.

Proof. See Exercise 1.1. ■

Definition 1.1.5 — Affine, conic and convex combination. Let $a_1, \ldots, a_k \in \mathbb{R}^n$. For $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$, a linear combination $\sum_{i=1}^k \lambda_i a_i$ is called:

- an *affine combination* if $\sum_{i=1}^k \lambda_i = 1$,
- a *conic combination* if $\lambda_i \geq 0$ for all $i \in [k]$,
- a *convex combination* if it is both affine and conic, i.e., $\lambda_i \in [0,1]$ and $\sum_i \lambda_i = 1$.

Similarly, we define an affine, a conic, and a convex hull as a set of all affine, conic and convex combinations, respectively. It is described more formally in the following definition.

Definition 1.1.6 — Affine, conic, convex hull. Let $A \subseteq \mathbb{R}^n$. Then:

- the *affine hull* of A, denoted as *aff(A)*, is the set of all finite affine combinations of A,
- the *conic hull* of A, denoted as *cone(A)*, is the set of all finite conic combinations of A,
- the *convex hull* of A, denoted as *conv(A)*, is the set of all finite convex combinations of A.

The concept of convexity can be extended to functions in the following way:

Definition 1.1.7 — Convex function. Let $X \subseteq \mathbb{R}^n$ be a convex set. A function $f : X \to \mathbb{R}$ is *convex* if for all $x, y \in X$ and $\lambda \in [0,1]$ it holds that

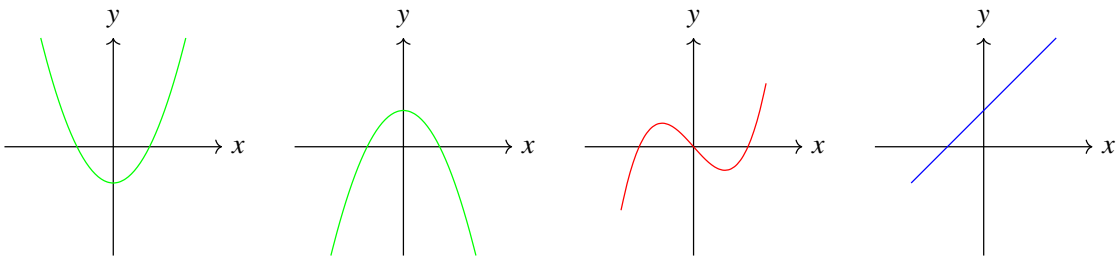
$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y).$$

If the above inequality is strict, the function is called *strictly convex*.

Moreover, if a function f is such that $-f$ is convex, respectively strictly convex we call it *concave*, respectively *strictly concave* function.

■ **Example 1.1.8** Below are four examples of functions, the one on the left is a strictly convex function, the second one is a strictly concave function as it is the first function times minus one. The third one is a neither convex nor concave function. The last function on the right is both convex and concave but is neither strictly convex nor strictly concave.

what is strictly convex?



■

There is a strong connection between convex sets and functions. In order to explore this we need the following notion.

Definition 1.1.9 — Epigraph. Let $X \subset \mathbb{R}^n$ be a set and $f : X \to \mathbb{R}$ a function on X. The *epigraph* of f is the set

$$\textbf{epi}\, f = \{(x,t) \in \mathbb{R}^{n+1} \mid x \in X, t \geq f(x)\}.$$

So in less formal words the epigraph consists of all points lying above the graph of f.

Proposition 1.1.10 A function $f : X \to \mathbb{R}$ is convex if and only if **epi** f is a convex set.

Proof. Assume that the function f is convex. Let (x_1,t_1) and (x_2,t_2) belong to the **epi** f, i.e., $f(x_1) \leq t_1$ and $f(x_2) \leq t_2$. For any $\lambda \in [0,1]$ we show that $(\lambda x_1 + (1-\lambda)x_2, \lambda t_1 + (1-\lambda)t_2) \in$ **epi** f. Indeed, by convexity of f,

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (a-\lambda) f(x_2) \leq \lambda t_1 + (1-\lambda)t_2.$$

For the converse, consider two points (x_1,t_1) and (x_2,t_2) that lie on the boundary of **epi** f, i.e., $f(x_1) = t_1$ and $f(x_2) = t_2$. Since **epi** f is a convex set, thus for any $\lambda \in [0,1]$ we have

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda t_1 + (1-\lambda)t_2 = \lambda f(x_1) + (1-\lambda) f(x_2),$$

which concludes the proof. ■

R If $f : X \to \mathbb{R}$, for $X \subseteq \mathbb{R}^n$, is differentiable, it is convex if and only if for all $x, y \in X$

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

Intuitively, this means that the tangent hyperplane at each point lies entirely below the function.

If a function f is twice differentiable, then it is convex if and only if its Hessian, i.e., its second derivative $\nabla^2 f(x)$, at every point $x \in X$ is positive semidefinite:

$$\nabla^2 f(x) \succeq 0.$$

For $n = 1$, so $X \subseteq \mathbb{R}$, this reduces to the well known condition $f''(x) \geq 0$, for all $x \in X$, that is, the second derivative has to be positive.

When considering concavity, the same statements hold with reversed inequalities. Note that strict convexity respectively concavity does not imply strict inequalities, as for the first function in Example 1.1.8.

For convex function the following properties are true.

Proposition 1.1.11 For $X \subseteq \mathbb{R}^n$ convex, let $f_1, \ldots f_s : X \to \mathbb{R}$ be convex functions. Then

1. $\sum_{i=1}^s \lambda_i f_i$ is convex for all $\lambda_i \in \mathbb{R}_{\geq 0}$,
2. $\max\{f_i, \ldots, f_s\}$ is convex,
3. $f_1(f_2(x))$ is convex for f_1 being non-decreasing.

Proof. See Exercise 1.3 ■

1.2 Polytopes and Polyhedra

A fundamental building block of polyhedra is the notion of a *half-space* and a *hyperplane*.

Definition 1.2.1 — Hyperplane, half-space. Let $a \in \mathbb{R}^n \setminus \{0\}$ and $\beta \in \mathbb{R}$. The $(n-1)$-dimensional subspace $H = \{x \in \mathbb{R}^n : a^\top x = \beta\}$ is called a *hyperplane*.

Moreover a hyperplane H defines a positive and negative *half-space* $H^+ = \{x \in \mathbb{R}^n : a^\top x \geq \beta\}$ and $H^- = \{x \in \mathbb{R}^n : a^\top x \leq \beta\}$, respectively.

Note that the vector a is orthogonal to H. Hyperplanes naturally connect to convex sets through the concept of separation. The following theorem formalizes this concept.

Theorem 1.2.2 — Separating hyperplane theorem. Let $X, Y \in \mathbb{R}^n$ be two nonempty, disjoint convex sets. Then there exists $a \in \mathbb{R}^n \setminus \{0\}$ and $b \in \mathbb{R}$ such that:

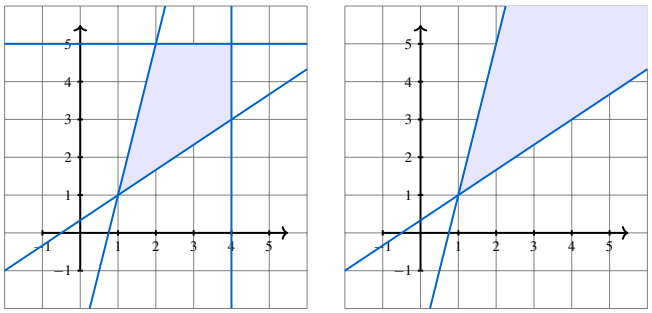
- $a^\top x \leq b$ for all $x \in X$,
- $a^\top y \geq b$ for all $y \in Y$.

The proof of the separating hyperplane theorem, Theorem 1.2.2 is omitted here. However, it can be found, for example, in [2, Section 2.5.1].

Now we are ready to define polyhedra and their bounded counterparts —polytopes.

Definition 1.2.3 — Polyhedron, polytope. Given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^n$ the set $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is called a *polyhedron*. Polyhedron P can alternatively be seen as the intersection of the finitely many half-spaces defined by the rows $a_i \in R^n$ of A and the vector entries b_i. A bounded polyhedron is called a *polytope*.

■ **Example 1.2.4** Example of a polytope, on the left and an unbounded polyhedron, on the right.



■

We introduce the following definitions for polyhedra that can be applied also for general subsets of $\mathbb{R}^n$.

Definition 1.2.5 Let $P \subseteq \mathbb{R}^n$. The set

$$\int(P) := \{x \in P : B(x, \varepsilon) \subseteq P, \text{ for some } \varepsilon > 0\}$$

is called the *interior* of the set P.

Definition 1.2.6 Let $P \subseteq \mathbb{R}^n$. The set

$$\text{relint}(P) := \{x \in P : B(x, \varepsilon) \cap \text{aff}(P) \subseteq P, \text{ for some } \varepsilon > 0\}$$

is called the *relative interior* of the set P.

Definition 1.2.7 — Supporting hyperplane. For a given set $P \subseteq \mathbb{R}^n$ a hyperplane H is called a *supporting hyperplane* if $P \cap H \neq \varnothing$ and either $P \subseteq H^+$ or $P \subseteq H^-$.

The following is the basic result connecting separating hyperplanes and convex sets.

Theorem 1.2.8 — Supporting hyperplane theorem. For any nonempty convex and closed set $S \subseteq \mathbb{R}^n$ and a point x belonging to the boundary of S, there exists a supporting hyperplane containing x.

Proof. We consider two cases. If the interior of S is nonempty, we apply the separating hyperplane theorem, Theorem 1.2.2, to the set $\{x\}$ and the interior of S. If the interior of S is empty, S must lie in the affine set of dimension $n-1$ or less, and any hyperplane containing that affine set contains S and x and is a supporting hyperplane. ■

R A partial converse of the supporting hyperplane theorem exists: If a closed, nonempty set $S \subseteq \mathbb{R}^n$ has a supporting hyperplane at every point x on its boundary, then S is convex.

We finish this section with a description of certain subsets of a polyhedron that deserve special attention because of their importance. We start with a formal definition of a dimension of a polyhedron.

Definition 1.2.9 The *dimension* of a polyhedron $P \subseteq \mathbb{R}^n$, denoted as $\dim(P)$, is the dimension of its affine hull $\operatorname{aff}(P)$, so:

$$\dim(P) := \min\{k \in \mathbb{Z}_{\geq 0} : \exists A \in \mathbb{R}^{n\times n},\ \operatorname{rank}(A) = n-k,\ Ax = Ay,\ \text{for all } x,y \in P\}.$$

Definition 1.2.10 — Face, vertex, edge, facet. Let $P \subseteq \mathbb{R}^n$ be a nonempty polyhedron, then:

1. a *face* of P is either P itself or the intersection of P with a supporting hyperplane,
2. a *vertex* of P is a 0-dimensional face of P,
3. an *edge* of P is a 1-dimensional face of P,
4. a *facet* of P is a $(\dim(P)-1)$-dimensional face of P.

R Every face of a polyhedron P is also a polyhedron of dimension $dim(P)$ or less.

Definition 1.2.3 gives an *inequality representation* of a polytope. The following proposition gives equivalent description of a polytope based on its vertices the *vertex representation*.

Proposition 1.2.11 — Vertex representation. Every polytope is the convex hull of its vertices.

R Note that the following converse holds: For every finite set $X \subseteq \mathbb{R}^n$, $conv(X)$ is a polytope.

A similar statement holds for unbounded polyhedra. For a proper description we introduce the definition of an extreme ray of a polyhedron.

Definition 1.2.12 Let $P \subseteq \mathbb{R}^n$ be a polyhedron. A point $r \in \mathbb{R}^n \setminus \{0\}$ is a *ray* of P if and only if

$$\{x + \lambda r |\ \lambda \geq 0\} \subseteq P$$

for some point $x \in P$.

Moreover, r is an *extreme ray* of P if it is not in a line segment between two distinct rays of P, i.e., there are no distinct rays r_1, r_2 (i.e., $r_1 \neq \mu r_2$ for any $\mu > 0$) of P and $0 < \lambda < 1$ such

that $r = \lambda r_1 + (1-\lambda) r_2$.

Proposition 1.2.13 — Minkowski Resolution Theorem. Every polyhedron is the Minkowski sum of the convex hull of its vertices and the conic hull of its extreme rays.

1.3 Cones

In the previous section, we studied polytopes and polyhedra. To better understand the connection between these sets, we present another fundamental class of sets, namely cones. As we will see later, cones deserve independent interest and will be used frequently in the following chapters.

Definition 1.3.1 — Cone. A set $K \subseteq \mathbb{R}^n$ is called a *cone* if for any $c \in K$ and $\lambda \in \mathbb{R}_{\geq 0}$ implies $\lambda c \in K$. Moreover a cone K is called:

- *convex cone* if it is convex,
- *solid cone* if it has nonempty interior,
- *pointed cone* if it contains no line, i.e., if $x, -x \in K \rightarrow x = 0$,
- *proper cone* if it is convex, closed, solid and pointed,
- *polyhedral cone* if it is a polyhedron.

R Any polyhedral cone K admits an inequality representation of the following form $K = \{x \in \mathbb{R}^n : Ax \leq 0\}$ for some matrix $A \in \mathbb{R}^{m \times n}$. Moreover the converse is true. That is, for every matrix $A \in \mathbb{R}^{m \times n}$ the set $K = \{x \in \mathbb{R}^n : Ax \leq 0\}$ is a polyhderal cone.

With this notation we can write 1.2.13 in an abstract way.

Proposition 1.3.2 For every nonempty polyhedron $P \subseteq \mathbb{R}^n$, there exist a polyhedral cone K and a polytope Q such that:

$$P = Q \oplus K.$$

Every cone has its dual cone. A dual cone is an interesting object used when we start discussing the optimization problems over cones. Now we define a dual cone and list its properties.

Definition 1.3.3 — Dual cone. Let $K \subseteq \mathbb{R}^n$ be a cone. The *dual cone* of K is the set defined as

$$K^* := \{l \in \mathbb{R}^n : \langle l, x \rangle \geq 0, \text{ for all } x \in K\}.$$

If $K = K^*$, the cone K is called *self dual*.

The upshot is that a vector is in the dual cone if and only if it is 0 or it defines a halfspace that completely contains the original cone.

For a pair of cone K and the dual cone K^* the following properties hold.

Proposition 1.3.4 Let $K_1, K_2 \subseteq \mathbb{R}^n$ be cones. Then

1. K_1^* is closed and convex (even if K_1 is not convex),
2. $K_1 \subseteq K_2 \Rightarrow K_2^* \subseteq K_1^*$,
3. K_1 is solid $\Rightarrow K_1^*$ is pointed. If $\overline{K_1}$ is pointed and convex $\Rightarrow K_1^*$ is solid,
4. $K_1^{**} = \overline{conv(K_1)}$, if K_1 is closed and convex, $K_1 = K_1^{**}$,
5. If K_1 is proper, then K_1^* is proper.

Proof. See Exercise 1.10 ■

1.4 Linear Programming

In this section, we discuss the most basic and the most heavily used class of convex optimization problems. In linear programming, the task is to minimize or maximize a linear objective function over a polyhedron.

Definition 1.4.1 — Linear Program (LP). Let $c \in \mathbb{R}^n$, $A, C, E \in \mathbb{R}^{m \times n}$ and $b, d, f \in \mathbb{R}^m$. An optimization problem is called a *linear program*, or short an *LP*, in a *general/standard/canonical form* if it is of the following form:

General :	*Standard* :	*Canonical* :
$\min c^\top x$	$\min c^\top x$	$\max c^\top x$
$Ax \geq b$	$Ax = b$	$Ax \leq b$
$Cx \leq d$	$x \geq 0$	$x \geq 0$
$Ex = f$		

We call an $x \in \mathbb{R}$ *feasible* if it fullfills the conditions of the LP.

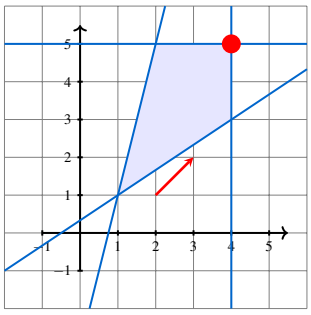
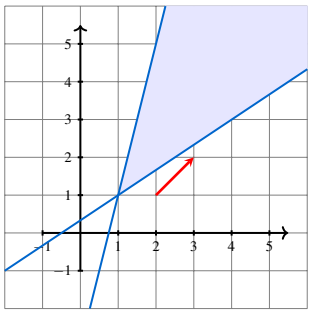
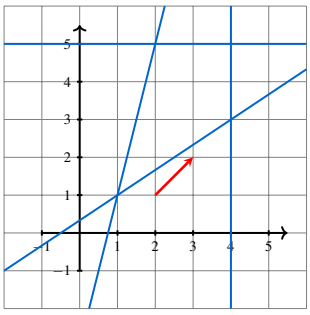
Note that even though the general form seems to be the most general and the most natural one, all of the above formulations are equivalent. Where, the equivalence is meant as the value of an optimal solution is the same for both problems, and an optimal solution for one problem can be transformed into an optimal solution for the second problem, and vice versa. That is a problem in general form can be rewritten to standard or canonical form and vice versa, see Exercise 1.11.

Definition 1.4.2 — Feasible, infeasible, unbounded LP. Given a linear program, w.l.o.g. in canonical form $\max\{c^\top x \mid Ax \leq b, x \geq 0\}$, we call it:

- *feasible* if it attains a finite optimum, i.e., if there is a feasible solution x such that for all feasible y we have $c^\top x \geq c^\top y$,
- *unbounded* if $c^\top x$ is unbounded over the feasible solutions, i.e., if for any C we find a feasible x with $c^\top x \geq C$,
- *infeasible* if it has no feasible solution at all.

■ **Example 1.4.3** Example of a linear program, on the left a feasible LP, in the middle an unbounded LP, and on the right an infeasible LP. The red arrow indicates the optimization direction and the red dot in the left image the optimal solution.

$\max x + y$	$\max x + y$	$\max x + y$
$4x - y \geq 3$	$4x - y \geq 3$	$4x - y \leq 3$
$3x - 2y \geq 1$	$3x - 2y \geq 1$	$3x - 2y \leq 1$
$x \leq 4$		$x \geq 4$
$y \leq 5$		$y \geq 5$

■

A fundamental concept for LPs is the concept of duality, in which we use the constraints of the LP to bound the value of an optimal solution. To motivate it consider the following example.

▪ Example 1.4.4

$$\begin{aligned} \max\ & 2x_1 + x_2 \\ & 2x_1 + x_2 \leq 4 \\ & x_1 + x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned} \qquad \begin{aligned} \max\ & 2x_1 + x_2 \\ & 2x_1 + x_2 \leq 4 \\ & x_1 + x_2 \leq 1 \\ & -x_1, -x_2 \leq 0 \end{aligned}$$

The first LP, which is in canonical form, can be modified slightly to take the form to the right. The first constraint is given by the same linear function as the objective. Thus for every feasible and in particular for the optimal solution, the objective value cannot exceed 4. To go even further, we can write

$$2x_1 + x_2 = 2(x_1 + x_2) + (-x_2)$$

which is a linear combination of the second and the third constraint. By plugging in the respective inequalities, we conclude that any feasible and thus also the optimal solution has an objective of at most $2 \cdot 1 + 0 = 2$. Indeed the feasible solution $(1,0)$ achieves this objective value of 2. So, in particular, we have just proven that it is an optimal solution. ▪

The idea of Example 1.4.4 was to express the objective as a linear combination of the constraints, which gives an upper bound on the optimal value. As every linear combination yields an upper bound we are interested in the minimal value this upper bound can attain. This can again be written as a linear program, the so-called dual linear program.

Here we introduce a dual variable for each constraint, corresponding to its factor in the linear combination we consider. As we want to express the objective function, each variable gives rise to an equality constraint. Furthermore, we have restricted to positive coefficients as otherwise, the inequalities of the constraints would flip, and we would not get an upper bound. Finally, the goal is to minimize the bound we obtain by applying the same linear combination to the right hand of the constraints. So the dual in case of Example 1.4.4 would be

$$\begin{aligned} \min\ & y_1 + y_2 + 0 \cdot y_3 + 0 \cdot y_4 \\ & 2y_1 + y_2 - y_3 - y_4 = 2 \\ & y_1 + y_2 - y_3 - y_4 = 1 \\ & y_1, y_2, y_3, y_4 \geq 0. \end{aligned}$$

This motivates the formal definition of the dual LP for the canonical form given below. For the standard form, similar reasoning holds, or alternatively, one can first transform the LP to its canonical form. The inequality in the dual may be unexpected at first, it comes from the fact that the dual variables of the positivity constraints can be seen as slack variables. To practice the notion of duality, try to use the motivation above to derive the forms below.

Definition 1.4.5 — Dual of an LP. Let $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The *primal* and *dual*

formulations of a linear program in a standard/canonical form are the following:

Standard		**Canonical**	
Primal :	*Dual* :	*Primal* :	*Dual* :
$\min c^\top x$	$\max b^\top y$	$\max c^\top x$	$\min b^\top y$
$Ax = b$	$c - A^\top y \geq 0$	$Ax \leq b$	$A^\top y \geq c$
$x \geq 0$		$x \geq 0$	$y \geq 0$

For these formal definitions, we can directly prove the bound we motivated in Example 1.4.4. This is called *weak duality* and we will obtain similar results for general convex optimization problems later (see Chapter 3).

Theorem 1.4.6 — Weak duality for linear programs. Let x, y be a feasible solution to the primal, dual formulation, respectively, of a linear program in a standard form. Then

$$c^\top x \geq b^\top y.$$

Proof. We have

$$c^\top x - b^\top y = x^\top c - (Ax)^\top y = x^\top (c - A^\top y) \geq 0,$$

where the first equality holds because of the equality constraint in the primal and the last inequality holds because of the inequality constraint in the dual. ■

R Clearly, by the solution of Exercise 1.11, an analogous relationship holds for the canonical form. In this case, using the notation from Definition 1.4.5, it holds that $c^\top x \leq b^\top y$.

Interestingly, for linear programs, an even stronger connection exists. We have seen in Example 1.4.4 that the optimal value of the dual LP agreed with the optimal value of the primal LP. This holds for general LP and is called *strong duality*. It is formalized in the following theorem.

Theorem 1.4.7 — Strong duality for linear programs. [16, Theorem 3.1] If both primal and dual are feasible, then their optimal costs are equal. That is, there exist feasible solutions x^*, y^* for primal and dual respectively such that $c^\top x^* = b^\top y^*$.

Note that, even though weak duality will hold for more general convex optimization problems that we will study soon (like semidefinite or conic programs), strong duality in an unconditional way holds exclusively for linear programs.

The following proposition summarizes the possible relationships between a primal and a dual formulation we obtain from Theorem 1.4.6 and Theorem 1.4.7.

Proposition 1.4.8 For a primal and a dual formulations of a linear program the following holds:

- Primal finite $\Rightarrow$ dual finite and their objective values agree,
- Primal unbounded $\Rightarrow$ dual infeasible,
- Primal infeasible $\Rightarrow$ dual unbounded or infeasible,

and, vice versa, as the dual of the dual LP is the primal LP (Exercise 1.12).

Proof. The first implication holds by strong duality, Theorem 1.4.7. The second one holds by weak duality, Theorem 1.4.6, and the last one holds because there exist pairs of primal—dual programs that are both infeasible. Indeed, by the second implication, consider a pair primal unbounded and dual infeasible. Construct a new linear program (with an arbitrary objective function), consisting of

both mentioned primal and dual constraints. This LP is infeasible and, by construction, its dual formulation is also infeasible. ■

To conclude the examination of dual LPs, we deduce another result that gives a more detailed insight into the balance between the primal–dual optimal solutions. In less formal words, we can conclude that for an optimal primal dual pair (x,y) either dual variable of a constraint is zero, or the corresponding constraint has to be tight (fulfilled with equality) for x.

Theorem 1.4.9 — Complementary slackness. Let x,y be feasible primal and dual solutions for linear program in the canonical form, respectively. Then x and y are optimal solutions if and only if

$$(b-Ax)^\top y = 0 \quad \text{and} \quad (A^\top y - c)^\top x = 0.$$

Proof. Following the proof of week duality for LPs in canonical form we have:

$$c^\top x \leq y^\top Ax \leq y^\top b$$

where, by strong duality Theorem 1.4.7, both inequalities are equalities which is equivalent to $(b-Ax)^\top y = 0$ and $(A^\top y - c)^\top x = 0$.

■

1.4.1 Algorithms for solving linear programs

In this section, we revise a few algorithms used to solve linear programs. We focus our attention on the Simplex method and the Ellipsoid method, give a brief overview of these algorithms' principles and discuss their properties both from a practical and theoretical perspective. For a much deeper insight into these methods, we refer the reader to [16, Chapter 2 and 4].

The Simplex method

The Simplex method was developed by Danzig in 1947 [5] and, up to this day, is the most famous and widely used method to solve linear programs.

To start the process, a feasible vertex solution is needed. This can be achieved by running the so-called phase I of the algorithm. Given a feasible vertex solution, the main idea behind this method is to travel along the edges of a polyhedron to better vertex solutions. This iterative process of improving the objective value is known as phase II of the algorithm and terminates after finitely many steps when the optimal vertex solution is found.

There are many different realizations of the idea presented above; however, for most variants, there are known examples for which the Simplex method needs exponential time to find an optimal solution. One such construction is the Klee-Minty cube [21] that was used to prove that the first version of the Simplex method discovered by Danzig [5] has worst-case exponential time performance [14].

Nevertheless, the Simplex method is very efficient and widely used in practice. In fact, many solvers up to this day use the Simplex method as the main tool to solve linear programs.

An interesting fact is, that the Simplex method was proved to have polynomial running time in the smoothed analysis [20]. Smoothed analysis, introduced in [20], interpolates between worst-case and average-case complexity. It measures the worst-case performance over inputs with small, usually Gaussian, random perturbation. The complexity is measured in terms of the encoding of the input and the magnitude of the perturbation, i.e., $1/\sigma$, where σ is the standard deviation of the Gaussian random perturbation.

The Ellipsoid method

The Ellipsoid method was first introduced by Show [18] and subsequently developed by Yudin and Nemirovski [24] and Shor [19] as a tool for solving convex optimization problems.

For the feasibility problems, the main idea behind this method is to enclose the feasibility region in an ellipsoid and iteratively establish if the center of the ellipsoid is a feasible point or not. If not, a hyperplane separating the center of the ellipsoid and the feasibility region is constructed that is further used to construct a smaller ellipsoid that contains the feasibility region. The process is continued until the volume of the ellipsoid is smaller equal to the volume of the feasibility region.

The ellipsoid method does not need to have direct access to the inequality description of the feasibility set by the principle described above, as the iterative process is based on a potentially simpler problem (called separation problem) of either deciding the feasibility of the point or finding a separating hyperplane.

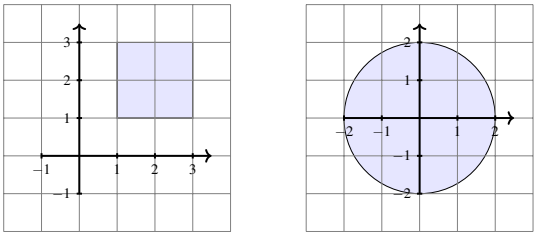
The method is known to be a polynomial time for linear programs [13], however, it is much slower in practice than the Simplex method. The number of iterations determines the running time of the method. This method can be used for other convex optimization problems, but not necessarily with polynomial running time. (e.g., for semidefinite programs). We refer to [15, chapter 4] for an in depth presentation.

1.5 Exercises

Exercise 1.1 Prove Proposition 1.1.4.

Exercise 1.2

1. Draw a Minkowski sum of the following two sets:



2. Construct two sets S and T (not necessarily convex) such that $S \oplus T$ is not convex. Can you find the simplest possible example?

Exercise 1.3

1. Prove Proposition 1.1.11.
2. Assume $f_1, f_2 : \mathbb{R} \to \mathbb{R}$ to be twice differentiable and apply the chain rule twice to $f_1(f_2(x))$. Give an alternative proof of the third property in 1.1.11.
3. Can you give an example of two convex functions f_1 and f_2 (non necessarily non-decreasing) such that $f_1(f_2(x))$ is not convex?
4. Does there exist an analogous of Property 3 in proposition 1.1.11 for concave functions?

Exercise 1.4 Show that the hyperbolic set $\{x \in \mathbb{R}^2_{\geq 0} : \ x_1 x_2 \geq 1\}$ is convex. As a generalization, show that $\{x \in R^n_{\geq 0} : \ \prod_{i=1}^n x_i \geq 1\}$ is convex.

Exercise 1.5 Let $\mathscr{S}^n$ be the set of real n by n symmetric matrices. Prove that the function $f : \ \mathscr{S}^n \to \mathbb{R}$ defined as $f(A) = \lambda_{\max}(A)$, that is, for a given symmetric n by n matrix A, f outputs its maximum eigenvalue, is a convex function.

Exercise 1.6 Prove that every polyhedron is a convex set.

Exercise 1.7 Prove that the nonnegative orthant in $\mathbb{R}^n$ is a cone. Which other properties from Definition 1.3.1 hold in this case?

Exercise 1.8 Prove that the nonnegative orthant in $\mathbb{R}^n$ is self dual.

Exercise 1.9 We define the *monotone nonnegative cone* as

$$K_{m+} := \{x \in \mathbb{R}^n | \ x_1 \geq x_2 \geq \cdots \geq x_n \geq 0\}.$$

1. Show that K_{m+} is a proper cone.
2. Find the dual cone of K_{m+}.

Exercise 1.10 — More difficult *. Prove Proposition 1.3.4.

Exercise 1.11 Given a problem in general form show how to write it in canonical and in standard form.

Exercise 1.12 Prove that for linear programs, both for a standard and a canonical form, the dual of the dual formulation is the primal formulation.

Exercise 1.13 — Programming Exercise. Consider the following (fictional) situation. A student plans his diet. He has to consume 2000kcal and 64g of vitamins per day. He can compose his diet from oats and apples. Oats have 300 kcal per 100g and 6g of vitamins, apples on the other hand have 200kcal and 10g of vitamins per 100g. The price of oats is 1CHF per kg while a kg apples is available for 2.5CHF.

The student wants to find the cheapest diet that satisfies his demands. First express this as a Linear Program. We want to solve it using CVXPY. The data can be found in the nutritions.txt file. This file contains as a first line the constraints. The second line lists the costs of the Foods. Finally there is one line per nutrition constraint in which the ammount each Food contributes to this constraint is listed.

Your program should first read in the data from this file and solve the corresponding linear program using CVXPY. Finally plot the feasible area of the LP, the optimal solution and the negative cost vector at the optimal solution.

You can find additional files with different constraint and Food combinations on the course website. You can test them as well. Note that the visualization will no longer work for scenarios with more than two (or strictly speaking more than three) types of Foods as the feasibility region will be high dimensional.

Exercise 1.14 — Programming Exercise. Assume you own a website on which you can show advertisements. There are several timeslots $t = 1, 2, \ldots, n$ and for each slot you know your sides traffic T_t. Furthermore you have contracts with m companies $i = 1, \ldots, m$ to show their ads and each one has to be shown an agreed minimum of c_i. For each visit on your page you can show at most one add. Let D_{it} denote the number of times you show add i in timeslot t. For the time being assume that D_{it} doesn't need to be integral. If the T_t are large this relaxation doesn't change the result significantly. (And of course your website is successful!)
Showing add i in slot t results with probability P_{it} in the visitor clicking it.

1. Assume you get a payment of p_i whenever ad i is clicked and you want to maximize the profit. Formulate this as an LP and solve it with CVXPY.
2. To be save from unexpected large payments the contracts contain a maximal amount B_i to be paid by company i. Show that this can be modeled as a concave objective function and thus can be maximized over. Implement it.
3. Now we go back to the original linear cost, but consider the more precise version where all D_i have to be integral. CVXPY can handle this when using

   ```
   cp.Variable(dimensions, integer=True)
   ```

   to generate the variables. Note that the resulting program is not a convex optimization problem. Observe how the runtime of your program changes. Do not worry if the bigger instances do not finish.

You can find example instances on the course website (with no adds being shown, we still work on the monetization) as well as a notebook containing a parser.

1.6 Solutions

Solution of Exercise 1.1

1. Let $p,q \in S \cap T$ and $\lambda \in [0,1]$. Since S is convex, $\lambda p + (1-\lambda)q \in S$, and similarly for T. This implies $\lambda p + (1-\lambda)q \in S \cap T$, so $S \cap T$ is convex.
2. Let $p,q \in W := \{Az+b : z \in S\}$ and $\lambda \in [0,1]$. There must exist two points $v,w \in S$ such that $Av+b = p$ and $Aw+b = q$. Since S is convex, $\lambda v + (1-\lambda)w \in S$. By the definition of W,

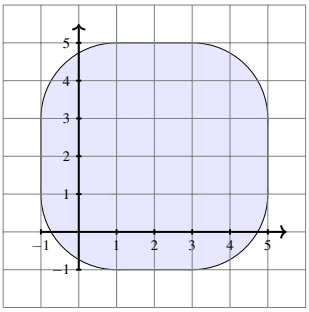
$$\begin{aligned} W \ni A(\lambda v + (1-\lambda)w) + b &= \lambda Av + (1-\lambda)Aw + b \\ &= \lambda Av + (1-\lambda)Aw + \lambda b + (1-\lambda)b \\ &= \lambda(Av+b) + (1-\lambda)(Aw+b) \\ &= \lambda p + (1-\lambda)q, \end{aligned}$$

   so W is convex since the parameters p,q,λ were chosen arbitrarily.
3. Let $p,q \in S \oplus T$ and $\lambda \in [0,1]$. There exist points $x_1,x_2 \in S$ and $y_1,y_2 \in T$ such that $x_1+y_1 = p$ and $x_2+y_2 = q$ by the definition of the Minkowski sum. We compute

$$\begin{aligned} \lambda p + (1-\lambda)q &= \lambda(x_1+y_2) + (1-\lambda)(x_2+y_2) \\ &= \underbrace{\lambda x_1 + (1-\lambda)x_2}_{\in S} + \underbrace{\lambda y_1 + (1-\lambda)y_2}_{\in T} \in S \oplus T. \end{aligned}$$

Solution of Exercise 1.2

1. The Minkowski sum is the following set:



2. Example: $S = \{\mathbf{0}\}, T = \{\mathbf{0},\mathbf{1}\}$.

Solution of Exercise 1.3

1. Proof of Proposition 1.1.11
   (a) Let $x,y \in X$ and $\mu \in [0,1]$, define $f = \sum_{i=1}^{s} \lambda_i f_i$, for $\lambda_i \in \mathbb{R}_{\geq 0}$. We compute

$$\begin{aligned} f(\mu x + (1-\mu)y) &= \sum_{i=1}^{s} \lambda_i f_i(\mu x + (1-\mu)y) \\ &\leq \sum_{i=1}^{s} \lambda_i \big(\mu f_i(x) + (1-\mu) f_i(y)\big) \\ &= \mu \sum_{i=1}^{s} \lambda_i f_i(x) + (1-\mu) \sum_{i=1}^{s} \lambda_i f_i(y) \\ &= \mu f(x) + (1-\mu) f(y) \end{aligned}$$

   by using convexity of the functions f_i and $\lambda_i \geq 0$ for the inequality. This implies that f is indeed a convex function.

(b) Let $x, y \in X$ and $\lambda \in [0,1]$, define $f = \max_{i\in[s]} f_i$. Let $j \in [s]$ such that $f_j(\lambda x + (1-\lambda)y) = f(\lambda x + (1-\lambda)y)$. By convexity of f_j, this implies

$$\begin{aligned} f(\lambda x + (1-\lambda)y) &\leq \lambda \underbrace{f_j(x)}_{\leq f(x)} + (1-\lambda)\underbrace{f_j(y)}_{\leq f(y)} \\ &\leq \lambda f(x) + (1-\lambda) f(y). \end{aligned}$$

(c) Let $x, y \in X$ and $\lambda \in [0,1]$. We have

$$f_1\big(f_2(\lambda x + (1-\lambda)y)\big) \leq f_1\big(\lambda f_2(x) + (1-\lambda) f_2(y)\big)$$

by convexity of f_2 and because f_1 is non-decreasing. Since f_1 is convex as well,

$$f_1\big(\lambda f_2(x) + (1-\lambda) f_2(y)\big) \leq \lambda f_1\big(f_2(x)\big) + (1-\lambda) f_1\big(f_2(y)\big).$$

Combining both inequalities implies that $f_1 \circ f_2$ is convex.

2. First we differentiate $f_1 \circ f_2$ once:

$$(f_1 \circ f_2)'(x) = f_1'(f_2(x)) f_2'(x).$$

Differentiating it a second time and using the chain and product rule gives us:

$$(f_1 \circ f_2)''(x) = f_1''(f_2(x))(f_2'(x))^2 + f_1'(f_2(x)) f_2''(x).$$

Now if f_1 and f_2 are both convex and f_2 is additionally monotonously increasing we have

$$f_1''(f_2(x)),\ f_1'(f_2(x)),\ f_2''(x) \geq 0.$$

Finally the last factor $(f_2'(x))^2$ is always non negative. So the assumptions together imply that $(f_1 \circ f_2)''$ is non-negative, which gives that $f_1 \circ f_2$ is convex.

3. If f_1 is not non-decreasing: Take for example $f_1 \colon x \mapsto x^2, f_2 \colon x \mapsto x^2 - 1$.
4. Yes: If f_1 and f_2 are concave and f_1 is non-decreasing, then $f_1 \circ f_2$ is concave as well. (The proof is similar to the convex case.)

Solution of Exercise 1.4

We start with proving that for $x_1, x_2 \geq 0$ and any $\lambda \in [0,1]$ we have

$$x_1^{\lambda} x_2^{(1-\lambda)} \leq \lambda x_1 + (1-\lambda) x_2.$$

To this end first note that if $x_1 = 0$ or $x_2 = 0$ the claim holds trivially. To prove other cases we use *Jensen's inequality* in its finite form. For a real convex function f, numbers $x_1, \ldots, x_n$ in the domain of the function f and positive weights $a_1, \ldots, a_n$, Jensen's inequality states that:

$$f\left(\frac{\sum_{i=0}^n a_i x_1}{\sum_{i=0}^n a_i}\right) \leq \frac{\sum_{i=0}^n a_i f(x_i)}{\sum_{i=0}^n a_i}.$$

We apply Jensen's inequality for convex function $f = -\log(x)$ and two positive numbers x_1, x_2 with corresponding weights λ and $1-\lambda$, for $\lambda \in [0,1]$. We get

$$-\log(\lambda x_1 + (1-\lambda)x_2) \leq -\lambda \log(x_1) - (1-\lambda)\log(x_2).$$

Taking exponents on both sides yields

$$x_1^{\lambda} x_2^{(1-\lambda)} \leq \lambda x_1 + (1-\lambda) x_2.$$

Now we prove that the hyperbolic set defined as

$$H_2 \coloneqq \{x \in \mathbb{R}^2_{\geq 0} |\ x_1 x_2 \geq 1\}$$

is convex. Let $p = (p_1, p_2) \in H_2$ and $q = (q_1, q_2) \in H_2$. For $\lambda \in [0,1]$, we want to prove that $\lambda p + (1-\lambda)q \in H_2$. Indeed, by the above proved inequality,

$$(\lambda p_1 + (1-\lambda)q_1)(\lambda p_2 + (1-\lambda)q_2) \geq p_1^{\lambda} q_1^{(1-\lambda)} p_2^{\lambda} q_2^{(1-\lambda)} = (p_1 p_2)^{\lambda} (q_1 q_2)^{1-\lambda} \geq 1,$$

where the last inequality holds since $p, q \in H$.

To prove that the generalized hyperbolic set H_n defined as

$$H_n \coloneqq \{x \in \mathbb{R}^n_{\geq 0} |\ \prod_{i=1}^{n} x_i \geq 1\}$$

is convex, we simply generalize the approach and for $p = (p_1, \ldots, p_n) \in H_n$ and $q = (q_1, \ldots, q_n) \in H_n$ and $\lambda \in [0,1]$. The claim follows form the inequalities

$$\prod_{i=1}^{n} (\lambda p_i + (1-\lambda)q_i) \geq \left(\prod_{i=1}^{n} p_i\right)^{\lambda} \left(\prod_{i=1}^{n} q_i\right)^{1-\lambda} \geq 1.$$

Solution of Exercise 1.5

First, we show that for any symmetric matrix $A \in \mathbb{R}^{n\times n}$ and any $x \in \mathbb{R}^n$ with $\|x\|_2 = 1$, we have $x^T A x \leq \sigma_{\max}(A)$. By the spectral theorem, A has an orthogonal eigenbasis, meaning that there exists an orthogonal matrix Q and a real diagonal matrix D such that $A = Q^T D Q$. Furthermore, the diagonal entries of D correspond to all eigenvalues of A. Let $v = Qx$. We have

$$x^T A x = x^T Q^T D Q x = v^T D v = \sum_{i=1}^{n} v_i^2 D_{ii} \leq \max_{i\in[n]} D_{ii} = \sigma_{\max}(A),$$

where the inequality follows from $\sum_{i=1}^{n} v_i^2 = \|v\|_2 = 1$ since Q is an orthogonal matrix.

We can now estimate $\sigma_{\max}(C)$, where C is the convex combination of any two symmetric matrices, i.e., $C = \lambda A + (1-\lambda)B$ for some symmetric matrices A, B and $\lambda \in [0,1]$. For this, let $z \in \mathbb{R}^n$ be a normed eigenvector of C corresponding to the eigenvalue $\sigma_{\max}(C)$. By the fact shown above,

$$\sigma_{\max}(C) = z^T C z = \lambda z^T A z + (1-\lambda) z^T B z \leq \lambda\, \sigma_{\max}(A) + (1-\lambda)\sigma_{\max}(B)$$

which shows that the function $\sigma_{\max}(\cdot)$ is indeed convex.

Solution of Exercise 1.6

Directly by Definition 1.2.3 a polyhedron P is a finite intersection of halfspaces. Since a halfspace is a convex set, Proposition 1.1.4 implies that P is convex.

For a direct proof, consider two points p, q which are in the polyhedron $P = \{x \in \mathbb{R}^n :\ Ax \leq b\}$ for some matrix $A \in \mathbb{R}^{m\times n}$ and some vector $b \in \mathbb{R}^m$. For any $\lambda \in [0,1]$, we have

$$A\big(\lambda p + (1-\lambda)q\big) = \lambda A p + (1-\lambda) A q \leq \lambda b + (1-\lambda) b = b,$$

meaning that $\lambda p + (1-\lambda)q \in P$. Hence, the polyhedron P is a convex set.

Solution to Exercise 1.7

The nonnegative orthant $\mathbb{R}_{\geq 0}$ is clearly a cone as for every $x \in \mathbb{R}_{\geq 0}$ and $\lambda \in \mathbb{R}_{\geq 0}$ we have $\lambda x \in \mathbb{R}_{\geq 0}$. Moreover, it is a polyhedral cone (defined by the intersection of halfspaces $x_i \geq 0$ for $i \in [n]$), thus is a convex cone). The cone is solid since for $\tilde{x} = (1, \ldots, 1)$ the ball $B(\tilde{x}, 1/2)$ centered in $\tilde{x}$ of radius $1/2$ belong to the cone, thus the cone has non-empty interior. It is a pointed cone, since for $x \in \mathbb{R}_{\geq 0}$ we have $-x \notin \mathbb{R}_{\geq 0}$. It follows that the cone is proper.

Solution to Exercise 1.8

Denote the nonnegative orthant of $\mathbb{R}^n$ as K. We want show that $K = K^*$.

$K \subseteq K^*$: For $x, y \in K$, $\langle x, y\rangle = \sum_{i=1}^n x_i y_i \geq 0$.

$K \supseteq K^*$: For the sake of contradiction, assume that exists $\ell \in K^*$ and $r \in \mathbb{N}$ such that $\ell_r < 0$. Let e^i be a unit vector with all zero entries but one in the $r'th$ entry. Then $\langle \ell, x\rangle = \ell_r < 0$, which is a contradiction, since $\ell \in K^*$.

Solution to Exercise 1.9

1. First we show that the cone K_{m+} is convex. Consider two points $x = (x_1, \ldots, x_n) \in K_{m+}$, $y = (y_1, \ldots, y_n) \in K_{m+}$. We want to prove that for any $\lambda \in [0,1]$ we have $\lambda x + (1-\lambda)y \in K_{m+}$. To this end we prove that for every $i \in [n-1]$ it holds $\lambda x_i + (1-\lambda)y_i \geq \lambda x_{i+1} + (1-\lambda)y_{i+1}$, which is trivially satisfied since $\lambda x_i \geq \lambda x_{i+1}$ and $(1-\lambda)y_i \geq (1-\lambda)y_{i+1}$.
   Next we note the cone is solid, as for $\tilde{x} = (1, 2, 3, \ldots, n)$ the ball $B(\tilde{x}, 1/2)$ centered in the point $\tilde{x}$ of radius $1/2$ is in K_{m+}, which shows that the interior is non-empty.
   Finally, the cone is pointed as for any $x \in K_{m+}$ the point $-x$ does not belong to K_{m+}. Thus by Definition 1.3.1 the cone K_{m+} is a proper cone.
2. Next we find a dual cone of K_{m+}. We use the following identity,

$$\begin{aligned}\sum_{i=1}^n x_i y_i =& (x_1 - x_2)y_1 + (x_2 - x_3)(y_1 + y_2) + (x_3 - x_4)(y_1 + y_2 + y_3) + \ldots \\ &+ (x_{n-1} - x_n)(y_1 + \ldots + y_{n-1}) + x_n(y_1 + \ldots + y_n)\end{aligned}$$

   that can be easily verified using term by term inspection. By Definition 1.3.3 we want to describe the set

$$K_{m+}^* = \left\{ y \in \mathbb{R}^n \middle| \sum_{i=0}^n x_i y_i \geq 0, \text{ for all } x \in K_{m+} \right\}.$$

   By the above identity, $y \in K_{m+}^*$ if and only if

$$\sum_{i=0}^n a_i \sum_{j=1}^i y_j \geq 0$$

   for any vector $(a_1, \ldots, a_n) \in \mathbb{R}^n_{\geq 0}$. This implies that the dual cone of K_{m+} can be written as

$$K_{m+}^* = \left\{ y \in \mathbb{R}^n \middle| \sum_{j=1}^k y_j \geq 0 \text{ for all } k \in [n] \right\}$$

Solution to Exercise 1.10

1. To check convexity, let $p, q \in K_1^*$ and $\lambda \in [0,1]$ and fix $x \in K_1$. By bilinearity of the scalar product, we have

$$\langle \lambda p + (1-\lambda)q, x\rangle = \underbrace{\lambda \langle p, x\rangle}_{\geq 0} + (1-\lambda)\underbrace{\langle q, x\rangle}_{\geq 0} \geq 0,$$

   meaning that $\lambda p + (1-\lambda)q \in K_1^*$. Since the choice of x is arbitrary, it implies that K_1^* is convex.
   To check that K_1^* is closed, fix $x \in K_1$ and let $(y_{(k)})_{k\in\mathbb{N}}$ be a Cauchy sequence in K_1^* with limit $y_\infty = \lim_{k\to\infty} y_{(k)}$. This implies that $\langle y_{(k)}, x\rangle$ for every $k \in \mathbb{N}$. Since the function $y \mapsto \langle y, x\rangle \geq 0$ is linear and thus continuous, we have

$$\langle y_{(\infty)}, x\rangle = \left\langle \lim_{k\to\infty} y_{(\infty)}, x \right\rangle = \lim_{k\to\infty} \underbrace{\langle y_{(k)}, x\rangle}_{\geq 0, \forall k \in \mathbb{N}} \geq 0$$

   Since the choice of x is arbitrary, this means that $y_{(\infty)} \in K_1^*$. We conclude that K_1^* is closed.

2. By definition, for every $y \in K_2^*$, we have $\langle y,x\rangle \geq 0$ for every $x \in K_2$ and in particular for every $x \in K_1$ as well,i.e.,

$$K_2^* = \{y|\ \langle y,x\rangle,\ \text{for } x \in K_2\} \subseteq \{y|\ \langle y,x\rangle,\ \text{for } x \in K_1\} = K_1^*.$$

   This implies that $y \in K_1^*$ which validates our claim.
3. First we prove that:

$$\text{if } K_1 \text{ is solid, then } K_1^* \text{ is pointed .}$$

   K_1 is solid implies that there exist a point $x \in K_1$ and $r > 0$ such that the ball $B(x,r)$ centered in x of radius r is in K_1. Let $z \in B(x,r)$. We want to prove that for any $y \in K^*$ if $-y \in K^*$ then $y = 0$. Note that if $y, -y \in K_1^*$ then $\langle y,z\rangle \geq 0$ and $\langle -y,z\rangle = -\langle y,z\rangle \geq 0$, which implies that $\langle y,z\rangle = 0$, for any $z \in B(x,r)$. It remains to prove that for any index $i \in [n]$, y_i=0. Indeed, consider a point $z = x + re^i$ where e^i is a unit vector with all zeros but one in r'th position, where it takes value 1. Since $\langle y,x\rangle = 0$ and $\langle y,z\rangle = 0$ we have $\langle y,z\rangle = \langle y, re^i\rangle = y_i = 0$ which finishes the proof.
   Next, let K_1 be a convex cone, we prove that:

$$\text{if } \overline{K_1} \text{ is pointed, then } K_1^* \text{ is solid.}$$

   First note that for any cone K, by Property 1, the cones K^* and $(\overline{K})^*$ are closed. Moreover, by Property 2, $(\overline{K_1})^* \subseteq K_1^*$ and by argument analogous to the proof of Property 1 $K_1^* \subseteq \overline{K_1}^*$, thus we have $K^* = (\overline{K})^*$. This implies that $K_1^* = \{y|\ \langle y,x\rangle \geq 0,\ \text{ for every } x \in \overline{K_1}\}$. It is easy to note that interior of K_1^* is $\operatorname{int} K_1^* = \{y|\ \langle y,x\rangle > 0,\ \text{ for every } x \in \overline{K_1} \setminus \{0\}\}$. Indeed, if for $y \in \operatorname{int} K_1^* \setminus \{0\}$ ($y = 0$ implies that there exists an $r > 0$, $B(0,r) \in K_1^*$ and thus $K_1^* = \mathbb{R}^n$ which is solid), and $x \in \overline{K_1} \setminus \{0\}$, we have $\langle y,x\rangle = 0$, it easy to find a point in a small ball around y such that $\langle y,x\rangle < 0$. Now we prove the claim. Since $\overline{K_1}$ is convex and pointed, there exist a hyperplane $H = \{x \in \mathbb{R}^n |\ a^\top x = 0\}$ such that $\overline{K_1} \subseteq H^+$ and $\overline{K_1} \cap H^+ = \{0\}$. This implies that for every $x \in \overline{K_1} \setminus \{0\}$ we have $\langle a,x\rangle > 0$ and thus $a \in \operatorname{int} K_1^*$, which implies that K_1^* is solid.
4. We prove that $K_1^{**} = \overline{\operatorname{conv}(K_1)}$.
   The inclusion $K_1^{**} \supseteq \overline{\operatorname{conv}(K_1)}$ is trivial. Indeed, for every $x \in K_1$ and $y \in K_1^*$, we have $\langle y,x\rangle \geq 0$, meaning that $K_1^{**} \supseteq K_1$. Since K_1^{**} is convex and closed, we have $K_1^{**} \supseteq \overline{\operatorname{conv}(K)}$. For the opposite inclusion, we can assume without loss of generality that K_1 is convex and closed, i.e. that $K_1 = \overline{\operatorname{conv}(K_1)}$. For the sake of contradiction, assume there exists a point $z \in K_1^{**} \setminus K_1$. This means that $\langle y,z\rangle \geq 0$ for every $y \in K_1^*$ but $\langle x,z\rangle < 0$ for some $x \in K_1$. By the *strong separating hyperplane theorem*, and because the set $\{z\}$ is compact there exists $a \in \mathbb{R}^n \setminus \{0\}$ and $b \in \mathbb{R}$ such that

$$\langle a,z\rangle < b \leq \langle a,x\rangle \quad \text{for every } x \in K_1.$$

   Because K_1 is a cone, we even have

$$\langle a,z\rangle < b \leq 0 \leq \lambda \langle a,x\rangle = \langle a, \lambda x\rangle \quad \text{for every } x \in K_1 \text{ and } \lambda \geq 0.$$

   Note that the latter inequality implies that $a \in K_1^*$, but $\langle a,z\rangle < 0$ means that $z \notin K_1^{**}$ which contradicts our assumption, hence $K_1^{**} \subseteq K_1$.
5. If K_1 is a proper cone, it is convex, closed, solid and pointed. By Property 1, K_1^* is closed and convex. Since K_1 is solid, by Property 3, K^* is pointed. Finally, since K_1 is pointed and convex, K_1^* is solid, and thus proper.

Solution to Exercise 1.11

In both cases, the key idea is to introduce so-called *slack variables*, which are token variables that have no other purpose than to modify the shape of linear (in-)equalities. For example, the inequality $2x_1 - 5x_2 \leq 3$ can be reformulated as an equality by adding a nonnegative variable $z \geq 0$:

$$2x_1 - 5x_2 \leq 3 \quad \Leftrightarrow \quad \begin{cases} 2x_1 - 5x_2 - 3 + z = 0; \\ z \geq 0. \end{cases}$$

Since the slack variables are omitted in the objective function, the optimal value of the problem remains unchanged.

Standard case:

We rewrite each variable x_i as $x_i = v_i - w_i$ and assume $v_i, w_i \geq 0$ for every $i \in [n]$. For the matrices A and C we also introduce two further sets of variables r_i, s_i for $i \in [m]$.

By such, the inequality $C_{j,1}x_1 + \cdots + C_{j,n}x_n \leq d_j$ is replaced by

$$C_{j,1}v_1 - C_{j,1}w_1 \pm \cdots + C_{j,n}v_n - C_{j,n}w_n + s_j = d_j$$

for every $j \in [m]$. The reformulation of the linear system $Ex = f$ is similar, except that no slack variable is needed. Since the expression $Ax \geq b$ is equivalent to $-Ax \leq b$, we reformulate the inequality $A_{j,1}x_1 + \cdots + A_{j,n} \geq b_j$ as

$$-A_{j,1}v_1 + A_{j,1}w_1 \mp \cdots - A_{j,n}v_n + A_{j,n}w_n + r_j = -b_j$$

for every $j \in [m]$. To summarize, the new standard-form LP is given as

$$\begin{aligned} \min(c_1v_1 - c_1w_1 \pm \cdots + c_nv_n - c_nw_n) \quad & \text{such that} \\ -A_{j,1}v_1 + A_{j,1}w_1 \mp \cdots - A_{j,n}v_n + A_{j,n}w_n + r_j &= -b_j \quad \forall j \in [m]; \\ C_{j,1}v_1 - C_{j,1}w_1 \pm \cdots + C_{j,n}v_n - C_{j,n}w_n + s_j &= d_j \quad \forall j \in [m]; \\ E_{j,1}v_1 - E_{j,1}w_1 \pm \cdots + E_{j,n}v_n - E_{j,n}w_n &= f_j \quad \forall j \in [m]; \\ v_i, w_i &\geq 0 \quad \forall i \in [n]; \\ r_j, s_j &\geq 0 \quad \forall j \in [m]. \end{aligned}$$

We can also summarize this in a vector-and-matrix representation as

$$\begin{aligned} \min l^T z \quad \text{such that} \quad Mz &= g; \\ z &\geq 0, \end{aligned}$$

with the vectors and matrices defined as follows. The variable vector is given as

$$z \coloneqq (v_1, \ldots, v_n, w_1, \ldots, w_n, r_1, \ldots, r_m, s_1, \ldots, s_m).$$

Similarly, the objective vector l is given as $l \coloneqq (c_1, \ldots, c_n, c_1, \ldots, c_n, 0, \ldots, 0)$ with its dimension matching z. The linear system of equality is provided by the block matrix and block column vector

$$M = \begin{pmatrix} -A & A & \mathbb{I}_m & 0 \\ C & -C & 0 & \mathbb{I}_m \\ E & -E & 0 & 0 \end{pmatrix} \quad \text{and} \quad g = \begin{pmatrix} -b \\ d \\ f \end{pmatrix}$$

respectively. Here, $\mathbb{I}_m$ denotes the $m \times m$-identity matrix.

Canonical case:

The procedure is similar to the standard case, except that no slack variables are required and that two modifications have to be made beforehand. First, the optimal value is found by the reformulation $\max c^T x = -\min -c^T x$, meaning that we take $-c$ as an objective vector instead of c, and that we take the negative of the computed optimum as the solution of the LP. Second, We replace the equality system $Ex = f$ with the inequality systems $Ex \leq f$ and $Ex \geq f$.

Solution to Exercise 1.12

This exercise is an immediate application of the techniques introduced in Exercise 2.7. We only consider the standard case. The canonical case can be proven similarly.

The approach is to reformulate the dual, i.e., the program $\max b^T y$ for $c - A^T y \geq 0$, in a primal form and to show that this program's dual is actually equivalent to the standard primal formulation. To do so, we split the variables y as $y_j = v_j - w_j$ with $v_j, w_j \geq 0$ for all $j \in [m]$ and introduce the slack variables r_i with $r_i \geq 0$ for all $i \in [n]$. We reformulate the dual program in block form as follows:

$$-\min \begin{pmatrix} -b \\ b \\ 0 \end{pmatrix}^T \begin{pmatrix} v \\ w \\ r \end{pmatrix} \quad \text{such that} \quad \begin{pmatrix} A^T & -A^T & \mathbb{I}_n \end{pmatrix} \begin{pmatrix} v \\ w \\ r \end{pmatrix} = c$$

and $v, w, r \geq 0$. The dual of this program is

$$-\max c^T z \quad \text{such that} \quad \begin{pmatrix} -b \\ b \\ 0 \end{pmatrix} - \begin{pmatrix} A \\ -A \\ \mathbb{I}_n \end{pmatrix} z \geq 0.$$

We notice that the matrix inequality is equivalent to the conditions $-Az \geq b$ and $-Az \leq b$ — both together being equivalent to $-Az = b$ — and $z \leq 0$. With the substitution $x := -z$, we recover the primal formulation since

$$-\max c^T z = -\max -c^T x = \min c^T x.$$

2. Convex optimization problems

2.1 Introduction

In this chapter, we define the main object of study in this book: convex optimization problems. They generalize linear optimization problems and are still solvable efficiently, for any desired accuracy. We present basic definitions, properties, and applications of convex optimization problems. The focus is on understanding the process of identifying, modeling, and solving this class of problems.

Next, we discuss a subclass of convex optimization problems, that is, conic optimization problems. Later in this book, we will construct several classes of optimization problems that generalize linear programs, and all of them belong to the class of conic programs. The main goal is to build a bridge between efficiently solvable but very restrictive, linear programs and difficult to solve but very general, mathematical optimization problems.

We start with the definition of mathematical optimization problems. We follow the definition in [2].

Definition 2.1.1 — Mathematical optimization problem. Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ for $i \in [m]$. A *mathematical optimization problem* in n variables has the form

$$\begin{aligned} &\min f(x) \\ &\quad g_i(x) \leq 0, \qquad \text{for } i \in [m], \\ &\quad x \in \mathbb{R}^n. \end{aligned}$$

The function f is called the *objective function* and the functions g_i are called *(inequality) constraint functions*.

R Linear programming is a subclass of mathematical optimization problems by defining $f(x) = c^\top x$ and $g_i(x) = a_i^\top x - b_i$ for $c, a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ for all $i \in [m]$.

In the following we define convex optimization problems as a subclass of mathematical optimization problems. Consider the following problem.

Definition 2.1.2 — Convex optimization problem. Let $f, g_1, \ldots, g_m : \mathbb{R}^n \to \mathbb{R}$ be convex functions. A *convex optimization problem* in general form is the following program:

$$\begin{aligned} &\min f(x) \\ &\quad g_i(x) \leq 0, \qquad \text{for } i \in [m]. \end{aligned}$$

Note that the feasibility region of a convex optimization problem is a convex set as it is an intersection of convex sets. Moreover, the following program

$$\min\{f(x) |\ g_i \leq 0,\ h_j(x) = 0,\ \text{for } i \in [m],\ j \in [\ell]\} \tag{2.1}$$

is a convex program if $g_i,\ \text{for } i \in [m]$ are convex functions and $h_j,\ \text{for } j \in [\ell]$ are affine functions. Here we use the the negative of an affine function is still affine and hence convex. Note that this is unique to affine functions.

R In some literature a convex program of the form in Equation (2.1) is called in standard form. Here we follow the more common convention of calling this a general form, which is consistent with the common convention used for linear programs.

In the following theorem, we prove the main property of convex programs. This theorem states for convex programs, every local minimum is a global minimum.

Theorem 2.1.3 Let $f :\ S \to \mathbb{R}$ be a convex function, for S being a convex set. Then every $\bar{x} \in S$ that is a local minimum of f over S is also a global minimum of f over the set S.

Proof. Let $\bar{x} \in S$ be a local minimum of f over the set S. Since $\bar{x}$ is a local minimum, there exists $r \in \mathbb{R}_{>0}$ such that $f(\bar{x}) \leq f(x)$ for all $x \in B(\bar{x}, r)$. For the sake of contradiction assume that $\bar{x}$ is not a global minimum of f over the set S. That is, there exists a point $\bar{y} \neq \bar{x}$, $\bar{y} \in S$ such that $f(\bar{y}) < f(\bar{x})$. We show that this leads to a contradiction. Indeed, let $\lambda \in [0,1]$ be such that $\lambda\bar{y} + (1-\lambda)\bar{x} \in B(\bar{x}, r)$. Such lambda always exists, an example is

$$\lambda = \frac{r}{\|\bar{y} - \bar{x}\|},$$

thus we have that $f(\lambda\bar{y} + (1-\lambda)\bar{x}) \geq f(\bar{x})$. However, by the convexity of function f, we know that

$$f(\lambda\bar{y} + (1-\lambda)\bar{x}) \leq \lambda f(\bar{y}) + (1-\lambda) f(\bar{x})$$

which, by the assumption that $f(\bar{y}) < f(\bar{x})$, gives

$$f(\lambda\bar{y} + (1-\lambda)\bar{x}) < f(\bar{x}),$$

which is a contradiction. ■

A follow up of the above theorem is a statement that shows that the set of local minima for a convex program is a convex set.

Theorem 2.1.4 Let $f :\ S \to \mathbb{R}$ be a convex function, for S being a convex set. Then the set of optimal solutions of the problem

$$\min\{f(x) :\ x \in S\}$$

is convex.

Proof. Let us denote the set of optimal solutions of the convex problem as X^*. If X^* is empty, then the claim follows immediately. If X^* is non empty, let $x, y \in X^*$. Note that x and y do not have to necessarily be different points. Since both x and y are in X^*, they are optimal values for the convex program. Since f is a convex function, for any $\lambda \in [0,1]$, the point $\lambda x + (1-\lambda)y$ is in S and we have

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) = f(x) = f(y)$$

where the equalities follow from optimality of x and y. Since x and y are in X^*, there is no point in S for which f has smaller value. We have

$$f(\lambda x + (1-\lambda)y) = \lambda f(x) + (1-\lambda)f(y) = f(x)$$

thus also $\lambda x + (1-\lambda)y$ is in X^*. ■

Convex problems cover a broad class of problems, including linear programs and many other classes that we will study in the next chapters. Special classes of convex problems include:

- Linear program (LP): if $f, g_1, \ldots, g_m$ are affine functions, that is, are of the form $f = c^\top x$ and $g_i = a_i^\top x - b_i$.
- Quadratic programs (QP): if $g_1, \ldots, g_m$ are affine functions and f a convex quadratic function.
- Quadratically constrained quadratic programs (QCQP): if all $f, g_1, \ldots, g_m$ are convex quadratic functions.
- Second order conic programs (SOCP): if f is an affine function and $g_i(x) = \|A_i x + b_i\|_2 + c_i^\top x + d_i$ for $i \in [m]$.
- Semidefinite programs (SDP): if f is an affine function and $g_i(x) = \lambda_{max}(A_i^0 + A_i^1 x_1 + \ldots + A_i^n x_n)$, where all matrices A_i^j are symmetric for $i \in [m]$ and $j \in [n]$, for $\lambda_{max}(\cdot)$ being the maximum eigenvalue of a symmetric matrix.

2.1.1 Abstract convex optimization problems

Convex optimization problems, as defined in Definition 2.1.2, consist in minimizing a convex function f over a convex set that is explicitly described by inequalities involving convex functions. Minimizing a convex function over a convex set without explicit inequality description by convex functions is called *abstract convex optimization problem*. It is important to note that an explicit inequality description is needed to solve convex optimization problems efficiently.

■ **Example 2.1.5** Consider the following program

$$\begin{aligned} \min\ & f(x) := x_1^2 + x_2^2 \\ & g(x) := x_1/(1+x_2^2)^2 \leq 0, \\ & h(x) := (x_1 + x_2)^2 = 0. \end{aligned}$$

Note that in this form, the problem is an abstract convex optimization problem, but it is not a convex optimization problem since function h is not an affine function and g is not convex. However, this program can be rewritten in the following form.

$$\begin{aligned} \min\ & f(x) := x_1^2 + x_2^2 \\ & \tilde{g}(x) := x_1 \leq 0, \\ & \tilde{h}(x) := x_1 + x_2 = 0. \end{aligned}$$

Note that the feasibility region is the same for both programs and the function $\tilde{h}$ is an affine function, and $\tilde{g}$ is a convex function. ■

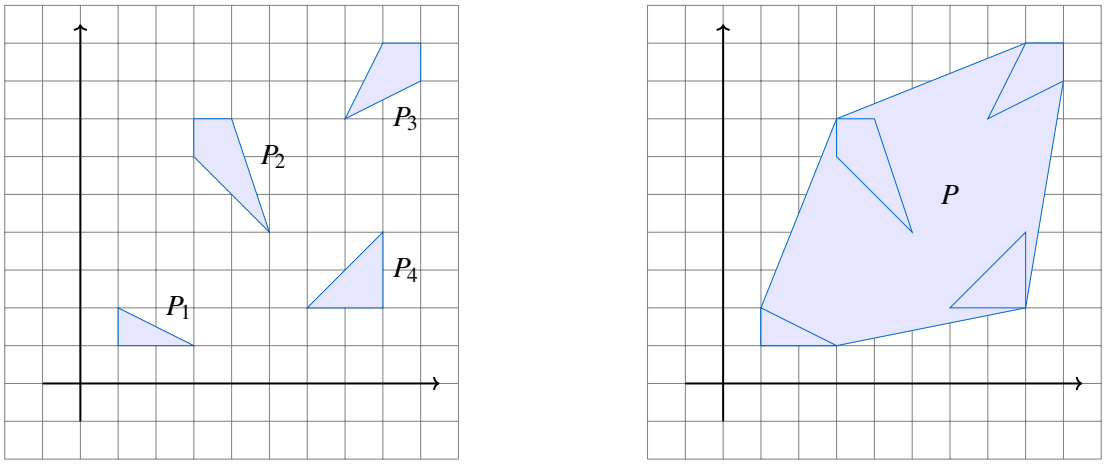
Describing a feasibility region by inequalities involving convex functions is an important step in formulating a problem as a convex optimization problem. To highlight the importance, we show another example known as disjunctive programming [1].

■ **Example 2.1.6** Consider a finite set of k polytopes $P_i \subseteq [0,1]^n$, for $i \in [k]$. We are interested in optimizing, possibly an affine function, over the convex hull of the union of polytopes $P_1, \ldots, P_k$, namely over the set P defined as

$$P = \operatorname{conv}\left(\bigcup_{i=1}^{k} P_i\right).$$

Since the convex hull of the union of $P_1, \ldots, P_k$ is a polytope, we are interested in its inequality representation.

Below, an example of 4 polytopes P_1, P_2, P_3, and P_4 in $\mathbb{R}^2$ on the left. The convex hull of the union of polytopes P_1, P_2, P_3 is on the right.



Let us try the most natural approach. Since any point in P is a convex combination of at most k points $\{x^1, \ldots, x^k\} \in \mathbb{R}^n$, where for $i \in [k]$, $x^i \in P_i$, we define the following set $\overline{P}$,

$$\overline{P} := \left\{ (x, x^1, \ldots, x^k, \lambda) \in \mathbb{R}^{n(k+1)+k} \;\middle|\; \begin{array}{rcll} x & = & \sum_{i=1}^{k} \lambda_i x^i & \\ \sum_{i=1}^{k} \lambda_i & = & 1 & \\ x^i & \in & P_i & \\ \lambda_i & \geq & 0 & \text{for } i \in [k] \end{array} \right\}.$$

First note that $\overline{P} \subseteq \mathbb{R}^{n(k+1)+k}$, but its projection on $\mathbb{R}^n$ defined by x (i.e. $(x, x^1, \ldots, x^k, \lambda) \mapsto x$) is indeed the polytope P. However, the issue is that $\overline{P}$ is not a polytope, as it is described by the function $(x, x^i, \ldots, x^k, \lambda) \mapsto x - \sum_{i=1}^{k} \lambda_i x^i$ which is not affine. To see this, note that both $\lambda_i \in \mathbb{R}$ and the vector $x^i \in \mathbb{R}^n$ are variables and thus $x = \sum_{i=1}^{k} \lambda_i x^i$ is not an affine function.

Fortunately, there is a way out of this issue, called *disjunctive programming*, introduced by Balas in [1]. The idea is simply to scale the polytopes P_i appropriately. For $i \in [k]$, $A_i \in \mathbb{R}^{m \times n}$ and $b_i \in \mathbb{R}^m$, suppose P_i admits the following inequality representation $P_i = \{x \in \mathbb{R}^n \mid A_i x \leq b_i\}$. For $\lambda_i \in \mathbb{R}$ let us define the scaled polytope $\lambda_i P_i$ as

$$\lambda_i P_i := \{x \in \mathbb{R}^n \mid A_i x \leq \lambda_i b_i\}$$

and $\tilde{P}$ as

$$\tilde{P} := \left\{ (x, x^1, \ldots, x^k, \lambda) \in \mathbb{R}^{n(k+1)+k} \;\middle|\; \begin{array}{rcll} x & = & \sum_{i=1}^k x^i & \\ \sum_{i=1}^k \lambda_i & = & 1 & \\ x^i & \in & \lambda_i P_i & \\ \lambda_i & \geq & 0 & \text{for } i \in [k] \\ x^i_j & \leq & \lambda_i & \text{for } i \in [k],\ j \in [n]. \end{array} \right\}$$

Note that, like $\overline{P}$, $\tilde{P} \subseteq \mathbb{R}^{n(k+1)+k}$, and its projection on $\mathbb{R}^n$ is exactly P. However, $\tilde{P}$ is a polytope as all the inequalities and equalities involve only affine functions. An important note is that the last set of inequalities of the form

$$x^i_j \leq \lambda_i \text{ for } i \in [k],\ j \in [n]$$

forces the x^i to be a zero vector whenever the corresponding λ_i is zero. The presented case, for all polytopes $P^i \in [0,1]^n$ for $i \in [k]$, can be extended to the general case by first transforming the inequality representation of every P^i to a standard form and then replacing the last type of inequality in $\tilde{P}$ with

$$x^i_j \leq d\lambda_i \text{ for } i \in [k],\ j \in [n]$$

where d is such that for every $i \in [k]$, $P^i \in [0,d]^n$ ∎

The above example shows how to build a linear program that optimizes over the convex hull of the union of a family of polytopes. We are interested in a more general setting in which we optimize a convex function over the convex hull of the union of convex sets described with inequalities involving convex functions. Note that this is a simple example of a transformation from an abstract convex program to a convex program. The problem is described more precisely in the Exercise 2.2.

2.1.2 Hidden convex optimization problems

In the previous subsection, we discussed problems that consist in optimizing a convex function over a convex set that does not have an explicit description with inequalities involving convex functions. In such cases, as shown in the Example 2.1.5, we can try to find an explicit description of the convex set to model the problem as a convex optimization problem.

In this subsection, we discuss an even more complicated situation in which the problem is not convex. However, there is a way to transform it into a convex optimization problem in some cases. The goal of this presentation is to show that many problems that seem to have non-convex nature can be cast as convex optimization problems. Such problems are called *Hidden convex problems*.

■ **Example 2.1.7** In this example we consider the *Trust Region Subproblem (TRS)* which is a well known non-convex problem that consists in minimizing a non-convex quadratic function subject to an Euclidean norm constraint of the form:

$$\min\{x^\top Ax + 2b^\top x + c :\ \|x\|^2 \leq 1\}$$

where $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Trust Region Subproblem is usually used as a subroutine in the *Trust Region Problem (TRP)*, which is an important method for solving nonlinear programs (that is not in the scope of this book that treats mostly about convex optimization problems).

Note that the problem is not necessarily convex since the quadratic objective function is not necessarily convex. In later chapters, we will characterize quadratic functions of that type based on the properties (eigenvalues) of matrix A. For now, we will give two examples of matrices that lead to convex and non-convex programs, respectively. First, let $n = 1$, $b = c = 0$ and let $A = 1$.

The resulting objective function is x_1^2, which is a convex function. However for $n = 2$ the following leads to a non-convex function: let $b = c = 0$ and let

$$A = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The corresponding objective function takes the form $x_1^2 - x_2^2$ which is a non-convex function.

We perform now a sequence of transformations of the TRS problem. First, note that since the matrix is symmetric, by spectral decomposition theorem, there exists an orthogonal matrix U (that is a matrix such that $U^\top = U^{-1}$) such that $A = UDU^\top$ for some diagonal matrix $D = \operatorname{diag}(d_1, \ldots, d_n)$. Orthogonal matrices have many interesting properties, one of which is that for any vector x we have $\left\|U^\top x\right\|^2 = \|x\|^2$, indeed:

$$\left\|U^\top x\right\|^2 = \left(U^\top x\right)^\top \left(U^\top x\right) = x^\top UU^\top x = x^\top x = \|x\|^2.$$

Thus, we can transform the TRS problem into:

$$\min\left\{x^\top UDU^\top x + 2b^\top UU^\top x + c : \left\|U^\top x\right\|^2 \leq 1\right\}.$$

By putting $y = U^\top x$, the problem reduces to

$$\min\left\{y^\top Dy + 2b^\top Uy + c : \|y\|^2 \leq 1\right\},$$

which, by denoting $e = U^\top b$, gives:

$$\begin{aligned} &\min \sum_i^n d_i y_i^2 + 2\sum_{i=1}^n e_i y_i + c \\ &s.t. \sum_{i=1}^n y_i^2 \leq 1. \end{aligned} \tag{2.2}$$

Note that the problem might still be non-convex since some of the diagonal entries might be negative. To transform the program into a convex form we need the following result.

Theorem 2.1.8 Let y^* be an optimal solution of (2.2). Then $e_i y_i^* \leq 0$ for all $i \in [n]$.

Proof. Let us denote the objective function of (2.2) by $f(y)$.We prove the claim for every $i \in [n]$ separately. For this reason fix an $i \in [n]$ and define the vector $\tilde{y} \in \mathbb{R}^n$ by

$$\tilde{y}_j = \begin{cases} y_j^*, & j \neq i, \\ -y_j^*, & j = i. \end{cases}$$

which is also a feasible solution of (2.2). Since y^* is an optimal solution, we get

$$f(y^*) \leq f(\tilde{y}).$$

This implies that

$$\sum_i^n d_i \left(y_i^*\right)^2 + 2\sum_{i=1}^n e_i y_i^* + c \leq \sum_i^n d_i \left(\tilde{y}_i\right)^2 + 2\sum_{i=1}^n e_i \tilde{y}_i + c$$

which after cancellation implies the claim that $e_i y_i^* \leq e_i \left(-y_i^*\right)$. ∎

By Theorem 2.1.8, for $e_i \neq 0$, we have that $\operatorname{sgn}(y_i^*) = -\operatorname{sgn}(e_i)$, where the sign function is defined as:

$$\operatorname{sgn}(y_i) = \begin{cases} 1, & y_i \geq 0, \\ -1, & y_i < 0. \end{cases}$$

For $e_i = 0$ the sign of y_i^* does not matter (for this to see note that both y_i^* and $\tilde{y}$ are optimal in the proof of Theorem 2.1.8). Thus we can make the change of variables $y_i = -\operatorname{sgn}(e_i)\sqrt{z_i}$ for some vector $z \in \mathbb{R}^n_{\geq 0}$. The TRS problem reduces to:

$$\begin{aligned} \min \ & \sum_i^n d_i z_i - 2\sum_{i=1}^n |e_i| \sqrt{z_i} + c \\ s.t. \ & \sum_{i=1}^n z_i \leq 1, \\ & z \in \mathbb{R}^n_{\geq 0}. \end{aligned}$$

which is a convex program since the objective is a convex function and the constraints are linear. ∎

Another example of transforming the non-convex program into a convex program will be presented in the chapter about Geometric Programs.

2.1.3 Optimization problem with equalities involving convex functions

As mentioned above the program in Equation (2.1) is a convex program if all $g_i,\ i \in [m]$ are convex and all $h_j,\ j \in [\ell]$ are affine functions. However, in some cases it is possible to solve a program where $h_j(x) = 0,$ for $j \in [\ell]$ are convex functions.

Proposition 2.1.9 Let $f, h, g_i,$ for $i \in [m]$, be convex functions. The program

$$\min\{f(x) |\ g_i \leq 0,\ h(x) \leq 0,\ \text{for } i \in [m]\} \tag{2.3}$$

is equivalent to the convex problem, where the equivalence is meant as the value of an optimal solution is the same for both problems, and an optimal solution for one problem can be transformed into an optimal solution for the second problem, and vice versa.

$$\min\{f(x) |\ g_i \leq 0,\ h(x) = 0,\ \text{for } i \in [m]\} \tag{2.4}$$

if at any optimal solution x^* of the convex problem we have $h(x^*) = 0$.

For more details regarding convex programs with equalities involving convex functions and its applications see Exercise 2.3 and 2.4.

2.1.4 Algorithms for solving convex programs

Like for linear programs, we do not cover the full details for algorithms solving convex optimization problems, as this is not in the scope of this study. Instead, we give a glimpse of intuition and requirements needed to solve convex optimization problems.

Convex optimization problems can be solved in polynomial time in the length of the problem encoding and the desired accuracy. For problems as in Definition 2.1.2 the *interior point method* can solve the problem for the case where functions $f, g_1, \ldots, g_m$ are twice continuously differentiable, and the problem is strictly feasible, i.e., there exist a point $x \in \mathbb{R}^n$ such that $g_i(x) < 0$ for every $i \in [m]$. This is equivalent to satisfying Slater's condition (discussed in more details in the chapter studying semi-definite programming, see Theorem 7.4.3), thus certifying the existence of the dual formulation. The idea behind the method is to construct barrier functions that encode the feasibility set. A particular example of a method to construct barriers is the Newton method that can be seen as

a technique for solving a linear equality constrained optimization problem, with twice differentiable objective, by reducing it to a sequence of linear equality constrained quadratic problems. For the full description of the method and the proof of its efficiency, see for example [2, Chapter 11].

2.2 Conic optimization problems

One of the simplest, yet very general realization of convex optimization problems are conic programs. In the following, analogously to the LPs, we define a conic program both in general and in standard form.

Definition 2.2.1 — conic program. Let $K \subseteq \mathbb{R}^n$ be a proper cone. Moreover, let $A, F \in \mathbb{R}^{m\times n}$, $b, g \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. A *conic program* has the following form:

$$\begin{aligned} &\textit{General}: && \textit{Standard}: \\ &\min\ c^\top x && \min\ c^\top x \\ &\quad -Fx - g \in K && \quad Ax = b \\ & && \quad x \in K. \end{aligned}$$

Note that standard form of a conic program has a nice geometric interpretation. For example, for linear programs, Definition 1.4.1, says that the feasibility set of the linear program is the intersection of the cone defined by the nonnegative orthant $\mathbb{R}^n_+$ and an affine subspace described by the equations $Ax = b$. We will see similar interpretations in the next chapters when studying semidefinite and second order conic programs.

The above definition covers only the case when a cone is a subset or the real vector space $\mathbb{R}^n$. Clearly, the definition can be extended to any finite inner vector space. To this end, we need the definition of a dual space.

Definition 2.2.2 — dual vector space. Given an inner vector space V over a field F, the *dual vector space* $V^*$is defined as the set of all linear maps $\phi : V \to F$.

A pair of an element in the dual vector space and an element in the inner vector space form a *pairing* $\langle\cdot,\cdot\rangle : V^* \times V \to F$ defined as $\langle\phi, x\rangle := \phi(x)$.

R A dual vector space V^* becomes itself a vector space over the field F when equipped with an addition and scalar multiplication satisfying:

$$\begin{aligned} (\varphi + \psi)(x) &= \varphi(x) + \psi(x), \\ (a\varphi)(x) &= a\,(\varphi(x)), \end{aligned}$$

for all ψ and $\phi \in V^*, x \in V$, and $a \in F$.

■ **Example 2.2.3** Consider the vector space over the reals consisting of two times two real matrices, $V = \mathbb{R}^{2\times 2}$. The trace of the matrix $A \in V$, $\mathrm{Tr}(A)$ is a linear functional $\mathrm{Tr} : V \to \mathbb{R}$ (as for $A, B \in V$ and $a, b \in \mathbb{R}$ it satisfies $\mathrm{Tr}(aA + bB) = a\,\mathrm{Tr}(A) + b\,\mathrm{Tr}(B)$) thus it belongs to the dual space, $\mathrm{Tr} \in V^*$. Note however, that this is not the entire dual space, it is just one element of the dual space V^*. ■

■ **Example 2.2.4** Consider a vector space V of n-variate real polynomials of degree at most d (note that this vector space is isomorphic to the vector space $\mathbb{R}^{\binom{n+d}{d}}$). Then $\phi : V \to \mathbb{R}$, defined as $\phi(p) = p(1)$ (which is a summation of the coefficients) is an element of the dual vector space V^*. Indeed, it is easy to check that ϕ is a linear functional satisfying $\phi(a\cdot p + b\cdot q) = a\phi(p) + b\phi(q)$ for every polynomial p and q and every $a, b \in \mathbb{R}$. ■

Now we are ready to define conic programs for general real vector spaces. Let S and T be two vector spaces over the reals and let $\mathscr{A}$ be a linear mapping from S to T, that is $\mathscr{A} : S \to T$. Since

every vector space over the field F has a dual vector space, let S^*, and T^* be the dual spaces of S and T respectively. Recall that for a mapping $\mathscr{A}$ there exists a unique *dual mapping* $\mathscr{A}^*: T^* \to S^*$ defined by:

$$\langle \mathscr{A}^* y, x\rangle_S = \langle y, \mathscr{A}x\rangle_T \qquad \text{for all } x \in S, y \in T^*.$$

For general real vector spaces, a conic program and its dual in standard form are defined in the following way.

Definition 2.2.5 — conic program duality in standard form. Given two real vector spaces S, T with scalar products $\langle\cdot,\cdot\rangle_S, \langle\cdot,\cdot\rangle_T$, a linear map $\mathscr{A}: S \to T$ and a proper cone $K \subseteq S$, for $b \in T$ and $c \in S^*$ the *primal* and the *dual* formulation of a conic program in standard form are the following.

Primal:

$$\begin{aligned} \min\ & \langle c, x\rangle_S \\ & \mathscr{A}x = b \\ & x \in K \end{aligned}$$

Dual:

$$\begin{aligned} \max\ & \langle y, b\rangle_T \\ & c - \mathscr{A}^* y \in K^* \end{aligned}$$

Theorem 2.2.6 — Weak duality for conic programs. Let x, y be a feasible solution to the primal, dual formulations respectively, of some conic program in standard form. Then

$$\langle c, x\rangle_S \geq \langle y, b\rangle_T.$$

Proof. We have:

$$\begin{aligned} \langle c, x\rangle_S - \langle y, b\rangle_T &= \langle c, x\rangle_S - \langle y, \mathscr{A}x\rangle_T \\ &= \langle c, x\rangle_S - \langle \mathscr{A}^* y, x\rangle_S \\ &= \langle c - \mathscr{A}^* y, x\rangle_S \\ &\geq 0, \end{aligned} \tag{2.5}$$

where the first equality holds because of the constraint in the primal formulation and the second one holds because of the definition of the dual mapping and the inequality holds because of the definition of dual cones. ■

Unlike for linear programming, strong duality (see Theorem 1.4.7) does not hold in general for conic programming. We will see in the next chapter an example of a conic (Semidefinite) program for which both primal and dual formulations are feasible, but attain different optimal values.

2.3 Miscellaneous

In this section, we discuss several interesting concepts regarding convex programs. We also look at some special cases of convex programs and analyze them. The goal is to give a broad perspective in identifying which problems can be solved and which cannot be solved by convex programs.

2.3.1 Quasiconvex programs

We start with a definition of a quasiconvex function.

Definition 2.3.1 — Quasiconvex function. Let $X \subseteq \mathbb{R}^n$ be a convex set. A function $f: X \to \mathbb{R}$ is *quasiconvex* if for all $x, y \in X$ and $\lambda \in [0,1]$ it holds that

$$f(\lambda x + (1-\lambda)y) \leq \max\{f(x), f(y)\}.$$

If the above inequality is strict, the function is called *strictly quasiconvex*.

Moreover, if a function f is such that $-f$ is quasiconvex, respectively strictly quasiconvex we call it *quasiconcave*, respectively *strictly quasiconcave* function.

The above definition leads to an extension of a definition of convex programs.

Definition 2.3.2 — Quasiconvex optimization problem. Let f be a quasiconvex function and let $,g_1,\ldots,g_m : \mathbb{R}^n \to \mathbb{R}$ be convex functions. A *quasiconvex optimization problem* in general form is the following program:

$$\begin{aligned} &\min\ f(x) \\ &\quad g_i(x) \leq 0, \qquad \text{for } i \in [m]. \end{aligned}$$

The main difference between convex programs and quasiconvex programs is that quasiconvex programs might have local optimal solutions that are not globally optimal solutions.

It is important to note that quasiconvex programs can be solved efficiently using the bisection over the family of convex programs. To show this, we start with defining an α-sublevel set.

Definition 2.3.3 The *α-sublevel set* of a function $f\colon\ \mathbb{R}^n \to \mathbb{R}$ is defined as:

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

The key observation is that if f is quasiconvex, then there exists a family of convex functions ϕ_t such that t-sublevel of f is 0-sublevel of ϕ_t, i.e.,

$$f(x) \leq t \quad \Leftrightarrow \quad \phi_t(x) \leq 0,$$

and for every x, $\phi_t(x)$ is a non-increasing function of t, i.e., $\phi_s(x) \leq \phi_t(x)$ for $s \geq t$.

■ **Example 2.3.4** Let $p,q\colon\ \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ be convex and concave function, respectively. The function $f(x) = p(x)/q(x)$ is a quasiconvex function. To family of functions $\phi_t(x)$ that satisfy the above properties can be constructed by

$$\phi_t(x) = p(x) - t \cdot q(x),$$

since

- $\phi_t(x)$ is a convex function for every $t \geq 0$,
- $p(x)/q(x) \leq t$ if and only if $\phi_t(x) \leq 0$.

■

To check if the optimal solution to the convex program is smaller equal to t it is enough to solve the following feasibility problem

$$\begin{aligned} &\text{find } x \\ &\quad \phi_t(x) \leq 0 \\ &\quad g_i(x) \leq 0, \qquad \text{for } i \in [m]. \end{aligned}$$

Moreover, if the problem is infeasible, the optimal value is larger than t.

This leads to the following algorithm to solve quasiconvex programs.

Input: $l \leq f^*$, $f^* \leq u$, and a tolerance $\varepsilon > 0$.
Output: $\hat{f}$ such that $\left|f^* - \hat{f}\right| \leq \varepsilon$.
repeat
Step 1: $t := (l+u)/2$
Step 2: solve the convex feasibility problem
Step 3: **if** feasible: $U := t$; **else** $l := t$
until $u - l \leq \varepsilon$.

2.3.2 Maximizing a convex function

One important point we want to make is that minimizing a convex function over a convex set is a tractable problem. However, maximizing a convex function over a convex set is more complex. One example of such a problem is minimizing the distance from a point to a polytope. Let us define the problem more formally.

Definition 2.3.5 — Min distance from a polytope.
Input: A point $x_0 \in \mathbb{R}^n$ and a polytope $P := \{x \in \mathbb{R}^n \mid g_i(x) \leq 0,\ i = 1,\dots,m\}$, for g_i being affine functions.
Output: A point $p_0 \in P$ that minimizes distance from x_0 to p_0.

The distance between two sets P and Q, in an Euclidean norm $\|\cdot\|$, is defined as

$$\mathbf{dist}(P,Q) = \inf\{\|x-y\| \mid x \in P,\ y \in Q\}$$

which is a convex function in the sense $x \mapsto \mathbf{dist}(x,Q)$.

Finding a minimum distance between the point and the polytope boils down to the following convex optimization problem

$$\begin{aligned} &\min \ \|x_0 - p\| \\ &s.t.\ g_i(p) \leq 0, \qquad \text{for } i = 1,\dots,m. \end{aligned}$$

When we change Definition 2.3.5 to maximize the distance, the problem is not that trivial. One way to deal with that is to use the property of maximizing convex functions over any set is that its maximum is the same as optimizing over the convex hull of that set. more precisely, let $S \in \mathbb{R}^n$ and let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function, then

$$\max_{x \in S} f(x) = \max_{x \in \operatorname{conv}(S)} f(x).$$

This implies that when we are given a vertex representation of a polytope P in the form $P = \operatorname{conv}\{p_1,\dots,p_K\}$ then we can simply solve the problem by

$$\max_{1 \leq i \leq K} \|p_i - x_0\|.$$

However, as pointed out in the previous chapter, vertex representation of a polytope is not always given. Moreover, a polytope described with polynomially many half-spaces might have an exponential number of vertices, which makes vertex representation not optimal in terms of encoding space.

2.3.3 Equality constraint elimination

For some cases a convex problem has to be written in the form without equality constraints. To do so, we can rewrite the constraint $Ax = b$ into $x = x_0 + Nz$, where x_0 is a particular solution to the equality constraints and the columns of N span the nullspace of A. Then we can rewrite the problem

$$\begin{aligned} &\min f(x) \\ &\quad g_i(x) \leq 0, \qquad \text{for } i \in [m], \\ &\quad Ax = b, \\ &\quad x \in \mathbb{R}^n. \end{aligned}$$

as

$$\begin{aligned} \min\ & f(Nz+x_0) \\ & g_i(Nz+x_0) \leq 0, \qquad \text{for } i \in [m], \\ & z \in \mathbb{R}^n. \end{aligned}$$

One remark is that, in practice, this is not always very efficient. The reason is that, for example, if A is a sparse matrix, some algorithms might exploit the sparsity. On the other hand the problem after the transformation, meaning the matrix N, might not inherit the sparsity characteristics.

2.4 Applications

In this section, we study some applications of convex optimization problems. The first application we want to discuss is the *Support Vector Machine (SVM)* model.

The SVM technique belongs to the class of supervised machine learning models that can solve classification and regression problems. Before discussing the SVM technique in detail, we use a more straightforward example to motivate the classification problem it solves.

Suppose we are given m points $\{x_1, x_2, \ldots, x_m\} \in \mathbb{R}^n$. Each point has a label $y_i \in \{-1, +1\}$ which indicates how the point is classified. An example data point can be 1024 by 1024 picture of a cat or a dog. In this case, with three RGB color channels, the picture can be seen as an $n = 1024 \cdot 1024 \cdot 3$ dimensional vector, and a label is -1 or 1 for cat or dog, respectively. Assume the points are linearly separable, which means that there exists a hyperplane separating points labeled with $+1$ from points labeled with -1. The goal is to find a hyperplane separating the points. The problem is formally defined in Definition 2.4.1.

Definition 2.4.1 — Linear classification problem.

Input: A set of m, n-dimensional points $\{x_1, x_2, \ldots, x_m\} \in \mathbb{R}^n$, each associated with a label $y_i \in \{-1, 1\}$.

Output: A hyperplane that strictly separates points labeled with $+1$ from points labeled with -1.

We attempt to solve the problem using mathematical programming methods. To do so, following the material we discussed in Section 1.2, we recall that a hyperplane H in $\mathbb{R}^n$ is defined by a tuple (w, b), where the vector $w \in \mathbb{R}^n \setminus \{0\}$ is called the normal vector of H, and $b \in \mathbb{R}$ is a scalar. The hyperplane H is the solution set of the equality

$$w^\top x + b = 0.$$

Note that in the linear classification problem the task is to find a hyperplane H characterized by a tuple (w, b) such that points labeled with 1 strictly belong to the positive halfspace, i.e., $H_+ \setminus H$ and the points labeled with -1 strictly belong to the negative halfspace, i.e., $H_- \setminus H$. In other words, we want to find a hyperplane (w, b) such that for every point labeled with 1 or -1 we have $w^\top x + b > 0$ or $w^\top x + b < 0$, respectively. Note however, that if a tuple $(w, b) \in \mathbb{R}^n \times \mathbb{R}$ defines a hyperplane H, then for every $c \in \mathbb{R} \setminus \{0\}$ the tuple $(c \cdot w, c \cdot b)$ also defines the hyperplane H. Thus, it is enough to find a hyperplane (w, b) such that for every point labeled with 1 or -1 we have $w^\top x + b \geq 1$ or $w^\top x + b \leq -1$, respectively. This leads to the following feasibility linear problem

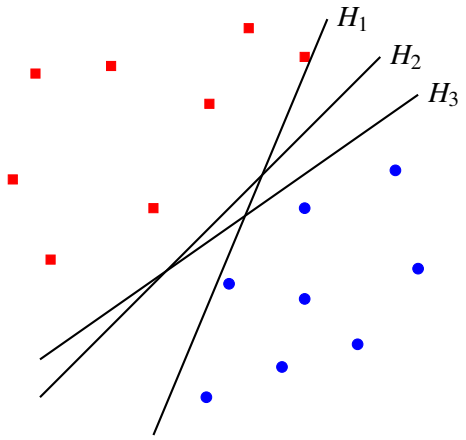
that solves the linear classification problem

$$\begin{aligned} &\min 0 \\ &\quad w^\top x_i + b \geq 1 \qquad \text{for every } i \in [n] \text{ s.t. } y_i = 1, \\ &\quad w^\top x_i + b \leq -1 \qquad \text{for every } i \in [n] \text{ s.t. } y_i = -1. \end{aligned}$$

Note that if the points are not linearly strictly separable, the above linear program does not have a feasible solution. If the points are linearly strictly separable, then the above linear program has many feasible solutions.

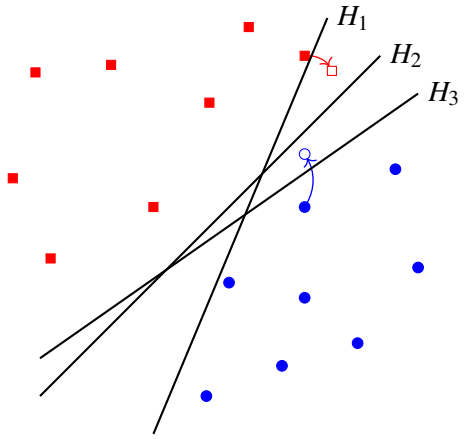
An example of a solution for the linear classification problem is shown in Figure 2.1.

Figure 2.1: An instance of the linear classification problem. Points labeled with -1 colored as red, points labeled with 1 colored as blue. Three possible solutions H_1, H_2 and H_3 are presented.



As shown in Figure 2.1 there are possibly many feasible solutions for this particular instance of the problem. The primary motivation for the Support Vector Machine method is to choose the best possible candidate hyperplane. Indeed, the main purpose of constructing classifiers in machine learning is to be able to classify unlabeled points outside the data set on which the classifier was constructed. In particular, if possible, a small perturbation of a labeled point should not lead to misclassification by our hyperplane. To visualize the situation, see Figure 2.2, where the positions of one red and one blue point were slightly perturbed. One can notice that this immediately leads to a wrong classification by hyperplanes H_1 and H_3.

Figure 2.2: An instance of the linear classification problem. Perturbation of the location of a sample red and a sample blue point lead to a misclassfication by hyperplanes H_1 and H_3, respectively.



A possible solution is to select a separating hyperplane that maximizes the distance, i.e., the margin from the labeled data. This leads to the Support Vector Machine problem presented in Definition 2.4.2.

Definition 2.4.2 — Support vector machine.

Input: A set of m, n-dimensional points $\{x_1, x_2, \ldots, x_m\} \in \mathbb{R}^n$, each associated with a $\{-1, 1\}$ label y_i.

Output: A hyperplane that separates points labeled with $+1$ from points labeled with -1 with a maximum possible margin.

Before we show how to model the SVM method using convex programming, let us introduce additional notation. Let H be a separating hyperplane. The closest data points to H are called support vectors. The area near H, which does not contain any data points, is called the margin.

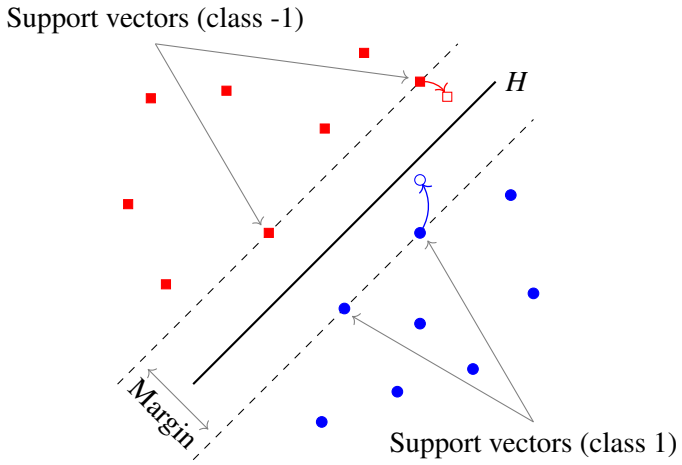


Figure 2.3: An instance of the linear classification problem. Hyperplane H is maximizing the margin, i.e., the distance to the closest point. Closest points are called support vectors. In this case, the hyperplane H is robust against small perturbation of the location of points.

Choosing the optimal hyperplane means choosing the one which has the biggest distance from all support vectors. Hence our goal is to find the hyperplane $w^\top x + b = 0$ with $w \in \mathbb{R}^n, b \in \mathbb{R}$ that maximize the margin.
As mentioned before, note that there are infinitely many pairs (w,b) such that the solution set of the equation $w^\top x + b = 0$ is the optimal hyperplane, e.g., for the given pair (w,b) and $c \in \mathbb{R} \setminus \{0\}$ the solution set of the equation $c \cdot w^\top x + c \cdot b = 0$ gives the same optimal hyperplane.
To finalize the example, we have to prove the following simple theorem.

Theorem 2.4.3 Assume that the points in the SVM instance are linearly separable. Then the following statements hold.

1. There always exists a pair (w,b) such that the hyperplanes containing the support vectors have equations $w^\top x + b = 1$ and $w^\top x + b = -1$, respectively.
2. For the pair (w,b) from point 1 the size of the margin is $\frac{2}{\|w\|}$.

Proof.

1. Let H, defined by $w^\top x + b = 0$, be the optimal hyperplane, and let $\{x_i, +1\}$ and $\{x_j, -1\}$ be the support vectors labeled with $+1$ and -1 respectively. The two supporting hyperplanes are parallel to H and passing through x_i or x_j. Let us call these two hyperplanes H_1 and H_{-1} respectively. Since H_1 is parallel to H, its equation differs from the one for H only by a constant. The same holds for H_{-1}. Then we can assume

$$H_1 : w^\top x + b = k_1$$
$$H_{-1} : w^\top x + b = k_{-1},$$

For some constants k_1 and k_{-1}. We also know that H has the same distance from H_1 and H_{-1} (why?), thus $k_1 = -k_{-1}$. Furthermore, the (w,b) that defines H is unique up to scaling. This means we can choose (w,b) such that $k_1 = 1$. Then the equations of the two hyperplanes are

$$H_1 : w^\top x + b = 1$$
$$H_{-1} : w^\top x + b = -1.$$

2. The margin is given by the distance between H_1 and H_{-1}. We know that $w \neq 0$ and notice that the vector w is orthogonal to the hyperplanes H, H_1 and H_{-1}. To prove that, it suffices to prove that $w^\top v = 0$ for every vector $v = x_1 - x_2$ where $x_1, x_2 \in H$. This implies, that $w^\top x_1 + b = 0$ and $w^\top x_2 + b = 0$. Then

$$w^\top v = w^\top (x_1 - x_2) = w^\top x_1 - w^\top x_2 = (-b) - (-b) = 0$$

which finishes the proof that w is orthogonal to H.
It remains to compute the distance between the hyperplanes H_{-1} and H_1. To this end, consider a point $x \in H_{-1}$, i.e. $w^\top x + b = -1$. Our goal is to compute $d \in \mathbb{R}$ such that the point $x + d\frac{w}{\|w\|}$ belongs to H_1, where $\frac{w}{\|w\|}$ is a unit normal vector of H_{-1}. Thus we get the following equality

$$w^\top \left(x + d\frac{w}{\|w\|} \right) + b = 1.$$

Since $w^\top x + b = -1$, we get that

$$d = \frac{2\|w\|}{w^\top w} = \frac{2}{\|w\|}.$$

■

Finally, we need to write a quadratic linear program to compute the best values for w and b. Notice that all points x_i labeled with $+1$ satisfy $w^\top x_i + b \geq 1$ and all points x_j labeled with -1 satisfy $w^\top x_j + b \leq -1$. Then all pairs (x_i, y_i) satisfy

$$y_i(w^\top x_i + b) - 1 \geq 0.$$

Moreover, we want to maximize $\frac{2}{\|w\|}$, yet this function is neither convex nor is the optimization direction correct. But observe that maximizing this function is the same as minimizing $\|w\|$, giving us a convex optimization problem. We can go even further and square this function and minimize $\|w\|^2$, which is a convex quadratic function. Note that squaring the objective preserves the minimal solution as the function $f(x) = x^2$ is monotone on the positve reals and the norm is always positve. Quadratic convex programs are often times faster to solve than a general convex one, which is the reason why we did the last step.
Hence the quadratic program to find the optimal solution is the following:

$$\begin{aligned} \min \quad & \|w\|^2 \\ \text{s.t.} \quad & y_i(w^\top x_i + b) - 1 \geq 0 \qquad \text{for each } i \in \{1, \ldots, m\}. \end{aligned}$$

Feature Maps

So far we investigated separating sets of points by a hyperplane. In many applications the classification is not that simple. For example take a look at Figure 2.4. There you can see two sets of points, in red and blue, that are following a nonlinear classification. We can still handle these with the theory we build up so far. The idea is to map the points to a higher dimensional space where they can be separated by a hyperplane.

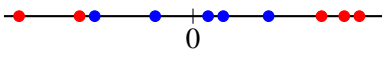


Figure 2.4: An instance of a nonlinear classification problem.

In this case we see for example that the absolute value of the points may play a role in their classification. A map that separates nicely when they differ in absolute value is for example $f\colon x \mapsto x^2$. In Figure 2.5 you can see the points $(x, f(x))$ for all original points x together with a separating line in this higher dimensional space.

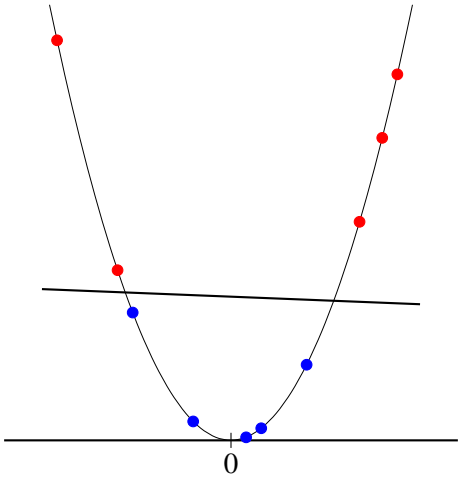


Figure 2.5: The instance mapped under the feature map $x \mapsto (x, x^2)$.

In general we are given a set of points $x_1, \ldots, x_m \in \mathbb{R}^n$ with labels y_i. A *feature map* is very general map $\Phi\colon \mathbb{R}^n \to \mathbb{R}^{n'}$. The goal is to choose a feature map such that the points $\Phi(x_1), \ldots, \Phi(x_n)$ are linearly separable. The big problem arising is how to choose the feature map. This has to be done for each application individually, exploiting the structure of the instance. Another problem is that the vector space in which we map might be very high, even infinite, dimensional. This makes finding a separating hyperplane in the modified instance hard or even computationally impossible. We will see a way to circumvent this when considering a dual version of SVM in chapter 3.

2.5 Exercises

Exercise 2.1 — Programming Exercise.

1. Use the convex program we constructed in section 2.4 to implement a SVM. Your program should accept a data file specifying two sets of points and return a separating hyperplane. Visualize the data set with two different colors and print the separator (Here you can assume the instance to be two dimentional). Also write a function that labels a new point x. On the webpage we give you some data files.
2. Extend your program such that it can use feature maps. You can find further test data for this on the webpage as well as a set of potential feature maps. You have to guess the right one. See the notebook for further details.

Exercise 2.2 — More difficult *. Let $f_{ij} : \mathbb{R}^n \to \mathbb{R}$ for $i \in [k]$, $j \in [m]$ be convex function. For $i \in [k]$ let

$$C_i := \{x \in \mathbb{R}^n | \ f_{ij} \leq 0, \text{ for } j \in [m]\}.$$

For a convex function $f_0 : \mathbb{R}^n \to \mathbb{R}$, we consider the following optimization problem.

$$\begin{aligned} &\min f_0(x) \\ &x \in \operatorname{conv}\left(\bigcup_{i=1}^{k} C_i\right). \end{aligned}$$

Formulate this problem as a convex optimization problem. Does an analogous construction to $\overline{P}$ from Example 2.1.6 work in this case? If not propose another approach.

Exercise 2.3 Prove that the program in (2.3) is equivalent to the convex program in (2.4) if there exists an index r such that:

- f is strictly monotonically increasing in x_r,
- $g_1, \ldots, g_m$ are nondecreasing in x_r,
- h is strictly monotonically decreasing in x_r.

Exercise 2.4 In this exercise, we show that the *optimal consumption* problem can be solved using convex programming. In this problem, an investor with an initial capital k_0 makes a sequence of decisions for periods $t \in \{0, \ldots, T\}$, where in every period the investor can decide an amount of money to spend on consumption. The remaining amount of money can be invested and, potentially, increase the capital for the next period. The goal is to maximize the total utility derived from consumption.

More precisely, let $c_t \geq 0$ be decision variables describing how much money an investor spends on consumption in period $t \in \{0, \ldots, T\}$. Let $0 < \beta < 1$ be a given *discount factor*. For an increasing concave function $u : \mathbb{R} \to \mathbb{R}$, let the total utility derived from the consumption, be

$$U := \sum_{t=1}^{T} \beta^t u(c_t).$$

Moreover, let $f : \mathbb{R} \to \mathbb{R}$ be an increasing concave function, called the *investment return function*. For $t \in \{0, \ldots, T\}$, let k_t denote the amount of money available for investment in period t. In every period $t \in \{0, \ldots, T\}$, the capital, not spent on consumption, can be invested so the following recursion is satisfied.

$$k_{t+1} = k_t + f(k_t) - c_t, \quad \text{for } t = 0, \ldots, T.$$

Show how to maximize U by solving a convex optimization problem. Explain how the problem you formulate is equivalent to the optimal consumption problem and how these two are related.

Exercise 2.5 Show that a linear program is an example of conic program by mapping standard form of a linear program from Definition 1.4.1 to standard form of a conic program.

Exercise 2.6 Show that the space of n-variate real polynomials of degree d or less, denoted as $\mathbb{R}[x]_d$, is isomorphic to $\mathbb{R}^{\binom{n+d}{d}}$.

Exercise 2.7 — Programming Exercise. A PNG image can be seen as an $m \times n$ matrix U^{orig} of pixels where each entry either denotes the intensity of that pixel, in the case of Greyscale images, or else a tuple of intensities for each color.

Consider the situation where we only received a corrupted image, so we only know the values of U^{orig} on a subset $\mathscr{K} \subset \{1,\dots,m\} \times \{1,\dots,n\}$ of the pixels. We want to restore the original image from this corrupted images as closely as possible. For this we want to find a matrix U that matches the known pixels on $\mathscr{K}$. Additionally the pixels we fill in shall be "close" to their neighbouring pixels. We measure this by the ℓ_2 variation of each pixel, which for the entry U_{ij} is defined as:

$$\left\| \begin{bmatrix} U_{i+1,j} - U_{ij} \\ U_{i,j+1} - U_{ij} \end{bmatrix} \right\|_2.$$

Implement this in python using CVXPY. You are given a corrupted image together with a mask that specifies which pixels are corrupted. The image may be Greyscale or use colors, the mask is saved as a Greyscale image, so as an array that is 0 if the pixel is corrupted and 1 if it is known. Your program should read in the corrupted image and the mask, then find a matrix that agrees with the corrupted image outside the mask and minimizes the total variation, i.e. the sum of the variation of all pixels. Finally plot your image. For the examples provided you can further compare your reconstructed image with the original.

You can find example images with masks on the course website as well as a colab notebook with which you can generate corrupted images from your own images. In both cases the corresponding masks are provided. In practice you could generate them from the knowledge, that you did not receive a pixel or you could mark the corrupted parts by hand.

Exercise 2.8 For optimal learning conditions, the ETH staff is currently working on a better light distribution in all (2 dimensional) lecture halls. Therefore they have declared the goal of providing every seat k, with a desired illumination level $I_k^* \in \mathbb{R}_{>0}$. Sadly, they quickly realised that it was not possible to achieve this exact goal and hence relaxed their intention to minimizing the deviation from the desired level. In order to achieve this objective efficiently, they want to create a convex optimization problem and need your help to do so.

Some background information that might help with the task. The light level at seat k provided by lamp j depends on the distance to the lamp, its power and the angle of reflection. More precisely, given the distance $r_{k,j}$ and angle $\theta_{k,j}$ it is proportional to the lamp power p_j with the factor $r_{k,j}^{-2} \max\{cos(\theta_{k,j}), 0\}$, decreasing with the square of the distance. The total illumination level at seat k, I_k is given by the sum of all involved lamps and the goal is to minimize the largest deviation in percent (i.e. $max_{k\in[n]} \left|\log(I_k) - \log(I_k^*)\right|$). Finally, each lamp has a maximal lamp power $p_j^{\max}$ which has to be taken into consideration. Note, that this problem is vaguely formulated on purpose, so there might be multiple correct solutions. Your convex optimization should however agree with common sense and the definition in the lecture notes.

Exercise 2.9 — Wire Spacing Problem (9). In a semiconductor chip, the n wires of a conductive layer run parallel. Stray capacitance between adjacent wires always occurs when the charge in one of the wires changes, as these are then behaving like capacitors. The (known) *switching* frequency of a wire w is modelled as a positive real number $\sigma = \sigma(w)$.

The switching is between the voltage 0 and the (constant) working voltage U. Neighbouring wires act as a capacitor, and to build up the corresponding electric field an energy is required that is directly proportional to its capacitance. This capacitance is proportional to the quotient of the

surface areas and the distance of the two involved lines. Assuming that the dimensions of the wires are fixed, the *current loss* of a wire w per time unit depends only on the distances d_{left} and d_{right} to its two neighbouring wires, in the form

$$\sigma(w)\left(\frac{1}{d_{\text{left}}}+\frac{1}{d_{\text{right}}}\right)$$

The goal is now to determine a spacing of the wires so that the occurring total stray capacity is minimized.

As constraints, a minimum tolerable distance $\delta \in (0,\infty)$ between any two wires and to the left and right edge of the chip is given (in order to avoid short-circuit faults). Also the maximal total width $\rho \in (0,\infty)$ of the chip is specified for the placement of the wires.

Formulate a Convex Optimization Problem which decides if the problem is feasible and if so, finds the optimal spacing given the number of wires n, the switching frequencies $\sigma_1,\ldots,\ldots,\sigma_n \in [0,\infty)$, the minimal tolerable distance $\delta \in (0,\infty)$ and the total width of the chip $\rho \in (0,\infty)$. You can ignore capacitances between non adjacent wires, since they are small compared to those between adjacent wires.

Exercise 2.10 — Handwritten digit recognition. Handwritten digit recognition with the MNIST dataset is nowadays mainly known as an introductory example for Convolutional Neural Networks. A basic approach to tackle it, is however based on Convex Programs.

The general goal in handwritten digit recognition is: given a greyscale image (28×28 pixels in this case) that represents a handwritten digit, decide which digit $d \in \{0,\ldots,9\}$ it is.

In order to learn how numbers are written, we are given a finite training set $P = \{(x,l)\} \subseteq \mathbb{R}^{784} \times \{0,\ldots,9\}$ where x is a given training image and l the digit it represents. Here we interpret each 28×28 image as a 784-dimensional vector by ordering the pixels in a fixed way. Now the goal is to find a matrix $D \in \mathbb{R}^{10x784}$ for which $y = Dx$ tells us which image x represents. An approach commonly used is to interpret y_i, for $i \in \{0,\ldots,9\}$, as the probability of x representing i. This does however not work immediately since y may have negative entries and the sum of entries might not be 1. Hence we need to normalize y with a function $z_i(y)$ such that $z_i(y) > 0$ (we need $>$ instead of $\geq$ in the following) and $\sum_{i=0}^{9} z_i(y) = 1$. To measure how *well* our matrix D is at recognizing digits, we use the loss function

$$\ell(D) = -\sum_{(x,l)\in P} \ln\left(z_l(Dx)\right).$$

which penalizes assigning a low probability to the correct value l.

Given this information,

1. find fitting functions $z_j\colon \mathbb{R}^{10} \to \mathbb{R}$ which satisfy $z_j(y) \in (0,1)$ and $\sum_{i=0}^{9} z_i(y) = 1$ for all $y \in \mathbb{R}^{10}$ and $i \in \{0,\ldots,9\}$.
2. prove that ℓ as a function $\mathbb{R}^{7840} \to (0,\infty)$ is convex for your choice of z_i or adapt z_i accordingly.
3. sum up that the problem can be solved with a Convex Optimization Problem.

Exercise 2.11 In this exercise we show that the following three convex problems are equivalent. please, explain how the solution of each problem is obtained from the solution of the other problems.

The problem data are the matrix $A \in \mathbb{R}^{m\times n}$ (with rows $a_i^\top$), the vector $b \in \mathbb{R}$, and the constant $M > 0$.

1. **The robust least-squares problem**

$$\min \sum_{i=1}^{m} \phi(a_i^\top x - b_i),$$

with variable $x \in \mathbb{R}^n$, where $\phi : \mathbb{R} \to \mathbb{R}$ is defined as

$$\phi(u) = \begin{cases} u^2 & \leq M, \\ M(2|u| - M) & |u| > M. \end{cases}$$

2. **The least-squares problem with variable weights**

$$\begin{aligned} \min \ & \sum_{i=1}^{m} \left(a_i^\top x - b_i\right)^2 / (w_i + 1) + M^2 \mathbb{1}^\top w \\ s.t. \ & w \geq 0. \end{aligned}$$

with variables $x \in \mathbb{R}^n$ and $w \in \mathbb{R}^m$, and domain $D = \{(x,w) \in \mathbb{R}^n \times \mathbb{R}^m : w > -1\}$.
Hint: Optimize over w assuming x is fixed, to establish a relation with the problem in part one.
(This problem can be interpreted as a weighted least-squares problem in which we are allowed to adjust the weight of the i'th residual. The weight is one if $w_i = 0$, and decreases if we increase w_i . The second term in the objective penalizes large values of w, i.e., large adjustments of the weights.)

3. **The quadratic program**

$$\begin{aligned} \min \ & \sum_{i=1}^{m} \left(u_i^2 + 2Mv_i\right) \\ s.t. \ & -u - v \leq Ax - b \leq u + v \\ & 0 \leq u \leq M\mathbb{1} \\ & v \geq 0. \end{aligned}$$

2.6 Supplement — Polynomial Optimization

In this section, we study the class of polynomial optimization problems. Polynomial optimization problem, in contrast to convex optimization problem, does not have to be convex. As an implication, polynomial optimization problems have high expressive power and can model a vast class of problems, including NP-hard problems arising in diverse fields of science and engineering. Hence, in general, we cannot solve polynomial optimization problems in polynomial time, unless P=NP. The goal of this section is to introduce basic definitions and properties of polynomial optimization problems, as an example of general class of problems for which convex optimization serves a main tool to construct efficiently solvable relaxations.

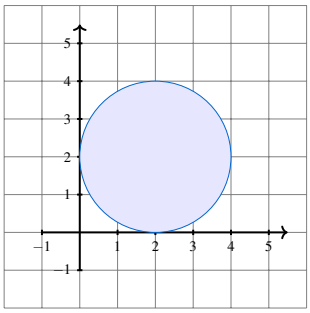
To this end, we consider the class of mathematical optimization problems as defined in Definition 2.1.1, and restrict the possible form of functions $f, g_1, \ldots, g_m$ towards introducing a polynomial optimization problem. Let $f, g_1, \ldots, g_m \in \mathbb{R}[x] := \mathbb{R}[x_1, \ldots, x_n]$. We start with the following definition.

Definition 2.6.1 — Basic closed semialgebraic set. Let $\mathcal{G} \subseteq \mathbb{R}[x]$ be finite subset of polynomials. The set $\mathcal{G}_+ \subset \mathbb{R}^n$ defined as

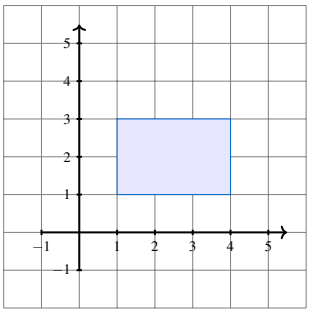
$$\mathcal{G}_+ := \{x \in \mathbb{R}^n | \; g(x) \geq 0, \text{ for all } g \in \mathcal{G}\}$$

is called a *basic closed semialgebraic* set generated by the set $\mathcal{G}$.

■ **Example 2.6.2** Examples of semialgebraic sets with their corresponding polynomials.



$g_1(x,y) = 4 - (x-2)^2 - (y-2)^2$



$g_1(x,y) = (x-1)(4-x)$

$g_2(x,y) = (y-1)(3-y)$

■

R Note that the description of a semialgebraic set is not unique, that is there exist sets $\mathcal{G}, \mathcal{G}' \subseteq \mathbb{R}[x]$ such that $\mathcal{G} \neq \mathcal{G}'$ but $\mathcal{G}_+ = \mathcal{G}'_+$. For example, adding a globally nonnegative polynomial to $\mathcal{G}$ does not modify the set $\mathcal{G}_+$.

■ **Example 2.6.3** The set on the right in Example 2.6.2 admits another description, for example with

$$\mathcal{G} = \{g_1(x,y) = x-1, \;\; g_2(x,y) = 4-x, \;\; g_3(x,y) = y-1, \;\; g_4(x,y) = 3-y\}.$$

■

Exercise 2.12 Consider the following two sets of polynomials $\mathcal{G}^1, \mathcal{G}^2 \subset \mathbb{R}[x,y]$:

$$\mathcal{G}^1 = \{4-(x-2)^2-(y-2)^2, \;\; -(x-1)(4-x), \;\; -(y-1)(3-y)\}$$

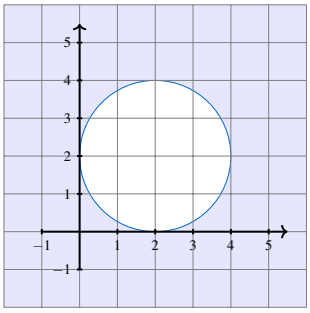
and

$$\mathscr{G}^2 = \mathscr{G}^1 \cup \{x^4y^2 + x^2y^4 - 3x^2y^2 + 1\}$$

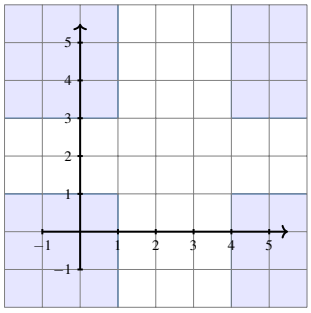
Prove that $\mathscr{G}^1_+ = \mathscr{G}^2_+$. ■

A critical property of basic closed semialgebraic sets is that they are not necessarily convex.

■ **Example 2.6.4** The following two sets are basic closed semialgebraic sets but they are not convex.



$g_1(x) = -4 + (x-2)^2 + (y-2)^2$



$g_1(x) = -(x-1)(4-x)$

$g_2(x) = -(y-1)(3-y)$

■

Definition 2.6.5 — Closed semialgebraic set. A finite union of basic closed semialgebraic sets in $\mathbb{R}^n$ is called a *closed semialgebraic set*.

R Semialgebraic sets are closed under finite unions, intersections, and complementation.

Moreover, semialgebraic sets are also closed under projections.

Theorem 2.6.6 — Tarski—Seidenberg theorem. Let X be a semialgebraic set in $\mathbb{R}^{n+1}$ and $\pi_n : \mathbb{R}^{n+1} \to \mathbb{R}^n$ be the projection defined as $\pi_n(x_1, \ldots, x_n, x_{n+1}) = (x_1, \ldots, x_n)$. Then $\pi_n(X)$ is a semialgebraic set in $\mathbb{R}^n$.

R An explicit inequality description of a projected semialgebraic set, obtained by Tarski—Seidenberg theorem, can be computed using the *Cylindrical Algebraic Decomposition*, introduced by Collins. The complexity of this algorithm is double exponential and it is tight.

We are ready to define polynomial optimization problems.

Definition 2.6.7 — Polynomial optimization problem. Let $f, g_1, \ldots, g_m \in \mathbb{R}[x_1, \ldots, x_n]$. A *polynomial optimization problem* in general form takes the form

$$\begin{aligned} \min\ & f(x) \\ & g_i(x) \geq 0, \qquad \text{for } i \in [m], \\ & x \in \mathbb{R}^n. \end{aligned}$$

Note that the set of constraints yields a feasibility set, and the value of an optimal solution depends only on the feasibility set, not on the description of the feasibility set. That is why polynomial programs can be rewritten using the notion of basic closed semialgebraic sets.

Proposition 2.6.8 Let, $f, g_1, \ldots, g_m \in \mathbb{R}[x_1, \ldots, x_n]$ as in Definition 2.6.7. Let $\mathcal{G} = \{g_0 = 1, g_1, \ldots, g_m\}$. A polynomial optimization problem can be equivalently written in the following form.

$$\begin{aligned} &\min f(x) \\ &\quad x \in \mathcal{G}_+. \end{aligned}$$

Polynomial programs that solve polynomial optimization problems have countless applications. They capture a very broad class of optimization problems. However, they are very difficult to solve in general. Mostly because polynomial optimization problems are not convex problems, in general.

■ **Example 2.6.9** Consider the following problem.

$$\begin{aligned} &\min f(x) := x^4 - 2x^2 + 1 \\ &\quad x \in \mathbb{R}^n. \end{aligned}$$

Note that $f(x) = x^4 - 2x^2 + 1 = (x^2 - 1)^2 \geq 0$ for all $x \in \mathbb{R}$. Moreover, for $x = -1$ and $x = 1$ we have $f(-1) = f(1) = 0$. However, $f(1/2 \cdot (-1) + 1/2 \cdot (1)) = f(0) = 1$, thus the function f is not convex. ■

Moreover, polynomial optimization problems are not only non-convex optimization problems, but they are also NP-hard, in general.

Exercise 2.13 Prove that solving a polynomial optimization problem is NP-hard. ■

Exercise 2.14 Give an example of a polynomial optimization problem that naturally arises in engineering or finance. ■

For this reason, most of the methods approach polynomial problems using convex relaxations. As we will see later on, these methods can provide a very efficient framework to tackle general polynomial problems.

2.7 Solutions

Solution of the Exercise 2.2

Like as in Example 2.1.6, we want to formulate the problem $\min\{f(x) : x \in C\}$, where $C := \operatorname{conv}\left(\bigcup_{i=1}^{m} C_i\right)$, as a convex program. We redefine the constraint functions f_{ij} as

$$\tilde{f}_{ij}(x,\mu) := \begin{cases} \mu f_{ij}\left(\frac{x}{\mu}\right), & \mu \neq 0; \\ 0, & \mu = 0, x = 0; \\ \infty, & \mu = 0, x \neq 0, \end{cases}$$

for $\mu \in \mathbb{R}_{\geq 0}$ and $x \in \mathbb{R}^n$ ($\tilde{f}_{ij}$ is sometimes called the *perspective of* f_{ij}). We now verify that $\tilde{f}_{ij}$ is convex. Let $x, y \in \mathbb{R}^n$, $\mu, \nu > 0$ and $\lambda \in [0,1]$ and let

$$\sigma = \frac{\lambda\mu}{\lambda\mu + (1-\lambda)\nu}.$$

We notice that $\sigma \in [0,1]$ and compute

$$\begin{aligned}
&\tilde{f}_{ij}(\lambda x + (1-\lambda)y, \lambda\mu + (1-\lambda)\nu) \\
&= \big(\lambda\mu + (1-\lambda)\nu\big) f_{ij}\left(\frac{\lambda x + (1-\lambda)y}{\lambda\mu + (1-\lambda)\nu}\right) \\
&= \big(\lambda\mu + (1-\lambda)\nu\big) f_{ij}\left(\sigma\frac{x}{\mu} + (1-\sigma)\frac{y}{\nu}\right) \\
&\leq \big(\lambda\mu + (1-\lambda)\nu\big)\left(\sigma f_{ij}\left(\frac{x}{\mu}\right) + (1-\sigma) f_{ij}\left(\frac{y}{\nu}\right)\right) \\
&= \lambda\mu f_{ij}\left(\frac{x}{\mu}\right) + (1-\lambda)\nu f_{ij}\left(\frac{y}{\nu}\right) \\
&= \lambda\tilde{f}_{ij}(x,\mu) + (1-\lambda)\tilde{f}_{ij}(y,\nu)
\end{aligned}$$

by using the convexity of f_{ij}. The cases $\mu = 0$ or $\nu = 0$ are trivial to verify. We define the scaled sets $\lambda_i C_i$ as in Example 2.1.6, i.e.

$$\lambda_i C_i := \{x \in \mathbb{R}^n : \tilde{f}_{ij}(x, \lambda_i) \leq 0 \ \forall j \in [m]\}$$

(notice that the choice $\tilde{f}_{ij}(0,0) = 0$ is necessary to make sure that $\lambda_i C_i \neq \emptyset$ for $\lambda_i = 0$) and

$$\begin{aligned}
\tilde{C} := \Big\{ &(x, x^1, \ldots, x^k, \lambda) \in \mathbb{R}^{(k+1)n} \times \mathbb{R}^k_{\geq 0} : \\
&x = \sum_{i=1}^{k} x^i, \sum_{i=1}^{k} \lambda_k = 1, x^i \in \lambda_i C_i \ \forall i \in [k] \Big\}.
\end{aligned}$$

It follows that the latter set is described by convex functions and its projection onto $\mathbb{R}^n$ is precisely C. This implies that the program

$$\begin{aligned}
&\min f_0(x) && \text{such that} \\
&\tilde{f}_{ij}(\lambda_i, x^i) \leq 0 && \text{for every } i \in [k], j \in [m]; \\
&\sum_{i=1}^{k} x^i = x; \\
&\sum_{i=1}^{k} \lambda_i = 1; \\
&\lambda_i \geq 0 && \text{for every } i \in [n],
\end{aligned}$$

is a convex optimization problem and equivalent to the initial problem $\min\{f(x) : x \in C\}$.

Solution of the Exercise 2.3

Let α be the optimal value of (2.3) and β that of (2.4). We have to show that $\alpha = \beta$ including the border cases $\{-\infty, \infty\}$.

For the first direction, $\alpha \leq \beta$ we observe that every x satisfying the constraints in (2.4) also satisfies the constraints of (2.3). This in particular implies that if (2.4) is unbounded or feasible so is (2.3).

For the other direction, $\alpha \leq \beta$ first observe that infeasibility of (2.3) implies infeasibility pf (2.4) so (2.3) is feasible and let x be a solution. We have to show, that that there is an y feasible for (2.4) such that $f(y) \leq f(x)$. Intuitively we want to reduce x_r. This does not increase f or any of the g_i by the conditions, yet it reduces h. One subtle point we have to be careful about is that this could fail to give a point with $h(y) = 0$. To exclude this we need to use the monotonicity of h.

So to do it formally for every $\eta \in \mathbb{R}_{\geq 0}$ we define y^η to be x in each component except for the r-th, there $y^\eta_r = x_r - \eta$. By the conditions on f and the g_i we have $f(y^\eta) \leq f(x)$ and $g_i(y^\eta) \leq g_i(x) \leq 0$. So we are done if we show, that there is an η with $h(y^\eta) = 0$. As h is convex and thus continuous it is enough to find an η with $h(y^\eta) \leq 0$.

For this we use that $h(y^1) > h(x)$. If $h(y^1)$ is already non negative, we are done so assume it is negative. Than for every $n \in \mathbb{N}$:

$$h(y^1) = h\left(\frac{n-1}{n}y^0 + \frac{1}{n}y^n\right) \leq \frac{n-1}{n}h(x) + \frac{1}{n}h(y^n).$$

So

$$h(y^n) \geq h(y^1) + (n-1)\left(h(y^1) - h(x)\right).$$

As $h(y^1) > h(x)$ this will surely be positive for n large enough. So there is an n with $h(y^n) \geq 0$ which gives an η with $h(y^\eta) = 0$. This y^η is feasible for (2.4) with value at most $f(x)$. So $\alpha \leq \beta$ as desired. In particular if (2.3) is feasible or unbounded so is (2.4).

Note that we needed strict monotonicity of h. If it is only monotone it could be a constant function, for example $h(x) = -1$. Then the two problems (2.3) and (2.4) are not always equivalent. As an example consider an arbitrary f, no g_i and $h(x) = -1$, then the first problem is feasible while the second is infeasible.

Solution of the Exercise 2.4

In this problem, the optimization variables are $c_0, \ldots, c_T, k_1, \ldots, k_T$ (recall that k_0 is given), and we formulate the optimization problem as follows:

$$\begin{aligned}
-\min -\sum_{i=0}^{T} \beta^i u(c_i) \quad & \text{such that} \\
c_s + k_{s+1} - k_s - f(k_s) \leq 0 \quad & \text{for every } s \in \{0, \ldots, T-1\}; \\
c_T - k_T - f(k_T) \leq 0; & \\
-c_i \leq 0 \quad & \text{for every } i \in \{0, \ldots, T\}; \\
-k_j \leq 0 \quad & \text{for every } j \in \{1, \ldots, T\}.
\end{aligned}$$

Since the function u is concave, $-u$ is convex and the (new) objective function $c \mapsto -\sum_{i=0}^{T} \beta^i u(c_i)$ is also convex(by an analogous property as presented in Proposition 1.1.11 for convex functions), and is obvious that the inequality constraint functions are convex as well by the same argument, since f is concave as well. The program above is therefore a convex optimization problem.

However the formulation above potentially allows a feasible solution with $k_{s+1} < k_s + f(k_s) - c_s$ for some $s \in \{0, \ldots, T-1\}$. In that case, take the same solution but replace c_s with $\tilde{c}_s = k_s + f(k_s) -$

$k_{s+1} > c_s$. It is trivial to verify that this new solution is feasible and that

$$\sum_{i=0}^{T} \beta^i u(c_i) \leq \sum_{\substack{i=0;\\ i\neq s}}^{T} \beta^i u(c_i) + u(\tilde{c}_s).$$

In particular, if the former point is an optimal solution then so is the latter, which means that the program above is indeed equivalent to the optimal consumption problem.

Solution of the Exercise 2.5
Solution of the Exercise 2.6
It is clear that the monomial basis, i.e. the set

$$\mathscr{B}_d = \left\{ x^\alpha := x_1^{\alpha_1} \cdot \cdots \cdot x_n^{\alpha_n} : \ \alpha \in \mathbb{N}_0^n, \alpha_1 + \cdots + \alpha_n \leq d \right\}$$

is a basis of $\mathbb{R}[x]_d$. Thus, it suffices to show that $|\mathscr{B}_d| = \binom{n+d}{d}$.

First we observe that every element of $\mathscr{B}_d$ can also be written as

$$x^\beta = 1^{\beta_0} \cdot x_1^{\beta_1} \ldots x_n^{\beta_n} \quad \text{with } \beta_0 + \beta_1 + \cdots + \beta_n = d.$$

In other words, $|\mathscr{B}_d|$ corresponds to the number of ways to place d balls into $n+1$ boxes. This can be illustrated by representing every monomial as a succession of dots and lines that represent the balls placed in the boxes and the delimitations between the boxes respectively, as in the following example:

$$\underbrace{\bullet\bullet}_{1^2} | \underbrace{\bullet\bullet\bullet\bullet}_{x_1^4} | \underbrace{\ }_{x_2^0} | \ldots | \underbrace{\bullet\bullet\bullet}_{x_n^3}$$

Every such representation consists of $n+d$ symbols (namely the d balls and n delimitating lines). Thus, to build such a sequence, one can choose d among the $n+d$ available slots, put all the balls in them and fill the remaining slots which lines, meaning that there are $\binom{n+d}{d}$ possible ways to do so. Since every sequence uniquely represents one monomial, this implies that there indeed $\binom{n+d}{d}$ different n-variate monomials of degree d or less.

Solution of the Exercise 2.8
The following is just one possible solution.
Using the notations from the exercise, let n denote the number of seats, m the number of lamps and

$$I_{k,j} := p_j r_{k,j}^{-2} \max\left\{cos(\theta_{k,j}), 0\right\}$$

be the light level at seat $k \in [n]$ provided by lamb $j \in [m]$. Then the problem can be formulated as

$$\begin{array}{lll} \min\limits_{p\in\mathbb{R}^m} & \max\limits_{k\in[n]} h\left(\dfrac{I_k}{I_k^*}\right) & \\ \text{s.t.} & -p_j & \leq 0 \\ & p_j - p_j^{\max} & \leq 0 \end{array}$$

where

$$I_k := \sum_{j=1}^{m} I_{k,j}, \text{ and}$$

$$h\colon (0,\infty) \to \mathbb{R}, x \mapsto \max\left\{x, \frac{1}{x}\right\}.$$

Here the constraint functions are linear hence convex and the objective is convex as a maximum of $2k$ convex functions and due to I_k being linear in p. To see that this problem is equivalent to our objective we first split our task into

$$|\log(I_k) - \log(I_k^*)| = \begin{cases} \log(I_k) - \log(I_k^*) & \text{, if } I_k \geq I_k^* \\ \log(I_k^*) - \log(I_k) & \text{, otherwise.} \end{cases}$$

Next we apply the exponential function. This gives us that the minimizer of $\left|\log(I_k) - \log(I_k^*)\right|$ is equal to the minimizer of

$$\begin{cases} I_k & \text{, if } I_k \geq I_k^* \\ \frac{1}{I_k} & \text{, otherwise,} \end{cases}$$

since exp is strictly increasing. Normalizing by I_k^* moves the breakpoint to 1, proving the claim.

Solution of the Exercise 2.9

We will start by modelling the problem as a general mathematical problem before showing that it is a convex problem. First of all we add $\sigma_0, \sigma_{n+1} = 0$ to deal with the borders and introduce variables $\xi_1, \ldots, \xi_{n+1}$. Here ξ_i describes the distance between wire $i-1$ and i, where 0 and $n+1$ correspond to the left and right border. Then the problem can be described as

$$\begin{aligned} \min_{\xi_1,\ldots,\xi_{n+1} \in \mathbb{R}} & \sum_{i=1}^{n} \sigma_i \left(\frac{1}{\xi_i} + \frac{1}{\xi_{i+1}} \right) \\ & \sum_{i=1}^{n+1} \xi_i \leq \rho \\ & \xi_i \geq \delta \quad \text{for } i \in [n+1]. \end{aligned}$$

which is infeasible iff $\rho > (n+1)\,\delta$ and produces an optimal solution otherwise. In order to see that the problem is convex, denote $\kappa_i := \sigma_{i-1} + \sigma_i$ for $i \in [n+1]$. Then we get the problem

$$\begin{aligned} \min_{\xi_1,\ldots,\xi_{n+1} \in \mathbb{R}} & \sum_{i=1}^{n+1} \frac{\kappa_i}{\xi_i} \\ & \sum_{i=1}^{n+1} \xi_i \leq \rho \\ & -\xi_i \leq -\delta \quad \text{for } i \in [n+1]. \end{aligned}$$

Now it is easy to see that the problem is convex, since sums of convex functions are convex and $x \mapsto \frac{1}{x}$ is convex on $(0,\infty)$. Since $\xi_i \geq \delta > 0$ and all the constraint functions are linear we get the claim.

Solution of the Exercise 2.10

First of all note, that due to the digits starting at 0 we also denote vectors with indices $0,\ldots,9$ instead of the commonly used 1 based indices in this exercise. We start with 1. and we will first show, that

$$z_j \colon \mathbb{R}^{10} \to \mathbb{R},\ z_j(y) := \frac{e^{y_j}}{\sum_{i=0}^{9} e^{y_i}}$$

fulfils the needed properties. Due to exp > 0 we get that $z_j(y) \in (0,1)$ and calculating

$$\begin{aligned}\sum_{j=0}^{9} z_j(y) &= \sum_{j=0}^{9} \frac{e^{y_j}}{\sum_{i=0}^{9} e^{y_i}} \\ &= \frac{\sum_{j=0}^{9} e^{y_j}}{\sum_{i=0}^{9} e^{y_i}} \\ &= 1\end{aligned}$$

yields the second statement.

For 2. we now need to check convexity of ℓ. Therefore we first note that

$$-\ln\left(z_l\left(Dx\right)\right) = \ln\left(\sum_{i=0}^{9} e^{(Dx)_i}\right) - \left(Dx\right)_l.$$

Next, note that $\left(\left(\lambda A + B\right)x\right)_i = \lambda\left(Ax\right)_i + \left(Bx\right)_i$, i.e. the second term is linear and hence concave. To take care of the first term, we look at the so called *LogSumExp* (LSE) function

$$\mathscr{L}\colon (0,\infty)^n \to \mathbb{R},\ \mathscr{L}\left(a_1,\ldots,a_n\right) \coloneqq \ln\left(\sum_{i=1}^{n} e^{a_i}\right).$$

There are different ways to show that $\mathscr{L}$ is convex. We will use Hölder's Inequality but there are also more basic ways as can be seen e.g. here. Hölder's Inequality says for $\frac{1}{p} + \frac{1}{q} = 1$ that

$$\sum_{k=1}^{n} |x_k y_k| \leq \left(\sum_{k=1}^{n} |x_k|^p\right)^{\frac{1}{p}} \left(\sum_{k=1}^{n} |y_k|^q\right)^{\frac{1}{q}}. \tag{2.6}$$

Now let $a, b \in (0,\infty)^n$, $\lambda \in [0,1]$ and define $p \coloneqq \frac{1}{\lambda}, q \coloneqq \frac{1}{\lambda - 1}$. This gives $\frac{1}{p} + \frac{1}{q} = 1$ hence we can use them for Hölder's. Therefore

$$\begin{aligned}\mathscr{L}\left(\lambda a + (1-\lambda)\, b\right) &= \ln\left(\sum_{i=1}^{n} e^{\lambda a_i + (1-\lambda) b_i}\right) \\ &= \ln\left(\sum_{i=1}^{n} \left(e^{a_i}\right)^{\frac{1}{p}} \left(e^{b_i}\right)^{\frac{1}{q}}\right) \\ &\overset{(2.6)}{\leq} \ln\left(\left(\sum_{i=1}^{n} e^{a_i}\right)^{\lambda} \left(\sum_{i=1}^{n} e^{b_i}\right)^{1-\lambda}\right) \\ &= \lambda \ln\left(\sum_{i=1}^{n} e^{a_i}\right) + (1-\lambda) \ln\left(\sum_{i=1}^{n} e^{b_i}\right),\end{aligned}$$

proving convexity. The final observation is that $f \circ h$ for convex f and affine h is again convex which yields that ℓ is convex.

Finally we can collect all pieces and end up with the (unconstrained) Convex Optimization Problem

$$\min_{D \in \mathbb{R}^{10\times 784}} \; -\sum_{(x,l)\in P} \ln\left(z_l\left(Dx\right)\right).$$

Solution of the Exercise 2.12

Since adding a globally nonnegative polynomial to $\mathscr{G}^1$ does not change the set $\mathscr{G}^1_+$, we show that this is indeed the case for the polynomial $h(x,y) = x^4y^2 + x^2y^4 - 3x^2y^2 + 1$.

First, we show that for $z_1, z_2, z_3 \geq 0$,

$$\log\left(\frac{1}{3}(z_1+z_2+z_3)\right) \geq \frac{1}{3}\big(\log(z_1)+\log(z_2)+\log(z_3)\big).$$

By concavity of the logarithm,

$$\begin{aligned}
\log\left(\frac{1}{3}(z_1+z_2+z_3)\right) &\geq \frac{1}{3}\log(z_1) + \frac{2}{3}\log(z_1+z_2) \\
&= \frac{1}{3}\log(z_1) + \frac{2}{3}\left(\log\left(\frac{1}{2}(z_2+z_3)\right)+\log(2)\right) \\
&\geq \frac{1}{3}\log(z_1) + \frac{2}{3}\log\left(\frac{1}{2}(z_2+z_3)\right) \\
&\geq \frac{1}{3}\log(z_1) + \frac{1}{3}\big(\log(z_2)+\log(z_3)\big).
\end{aligned}$$

By taking the exponent of the two sides of this inequality, we arrive at the inequality

$$\frac{z_1+z_2+z_3}{3} \geq \sqrt[3]{z_1 z_2 z_3}$$

which is known as the *arithmetic-geometric-mean inequality* (the general case with more than three summands can be shown similarly by induction).

Now, by the substitution $z_1 = x^4y^2$, $z_2 = x^2y^4$ and $z_3 = 1$, the arithmetic-geometric-mean inequality implies that $h(x,y) \geq 0$ for every $x, y \in \mathbb{R}$, meaning that

$$\mathscr{G}^2_+ = \mathscr{G}^1_+ \cap \underbrace{\big\{(x,y) \in \mathbb{R}^2 : \; h(x,y) \geq 0\big\}}_{=\mathbb{R}^2} = \mathscr{G}^1_+.$$

Solution of the Exercise 2.13

We prove that solving polynomial optimization problems is NP-hard in general. We show it by casting the Maximum Independent Set as a polynomial program.

Let $G(V,E)$ be an undirected graph, let $n = |V|$. An *independent set* of the graph G is a subset of vertices $S \subseteq V$ such that no two vertices in S are adjacent. Given graph G, the goal of the *Maximum Independent Set* problem is to find an independent set of maximum size. The problem is NP-hard and it is also NP-hard to approximate it with a factor $n^{1-\varepsilon}$ for any $\varepsilon > 0$ [11].

For every $v_i \in V$ let x_i be the binary variable with the intended meaning that it takes value 1 if the corresponding vertex is in the independent set and 0 it is not. The following program solves the Maximum Independent Set problem in graph G.

$$\begin{aligned}
\max \; & \sum_{i=0}^{n} x_i \\
& x_i + x_j \leq 1 \qquad \text{for all } (v_i, v_j) \in E \\
& x_i \in \{0,1\} \qquad \text{for all } v_i \in V
\end{aligned}$$

Where for every $v_i \in V$ the last constraint $x_i \in \{0,1\}$ can be rewritten in the following form:

$$x_i^2 - x_i \geq 0 \quad \text{and} - x_i^2 + x_i \geq 0$$

which results in a polynomial program.

Solution of the Exercise 2.14

3. Duality

In this chapter, we study the concept of duality for mathematical programs, in particular, for convex programs. We start with an intuitive reformulation of a generic mathematical program. The main idea we exploit in this chapter is to relax the reformulated problem into a family of problems, each of which is a lower bound for the original problem. Then we optimize over the family of relaxations to choose the one with maximal optimal value with the goal to obtain a possibly good lower bound on the optimal value of the original program. This approach is a cornerstone of the concept of duality.

3.1 Lagrange dual function and dual problem

We start with recalling the definition of the mathematical optimization problem as defined in Definition 2.1.1. For the sake of consideration in this chapter, we explicitly separate equalities and inequalities in the description of the mathematical program as follows.

$$\begin{aligned} \min\ & f(x) \\ & g_i(x) \leq 0, \qquad \text{for } i \in [m], \\ & h_j(x) = 0, \qquad \text{for } j \in [\ell], \end{aligned}$$

for $f : F \subseteq \mathbb{R}^n \to \mathbb{R}$, $g_i : G_i \subseteq \mathbb{R}^n \to \mathbb{R}$ for $i \in [m]$ and $h_j : H_j \subseteq \mathbb{R}^n \to \mathbb{R}$ for $j \in [\ell]$. The domain of the mathematical program we denote by $\mathbb{P} := F \cap \bigcap_{i=1}^{m} G_i \cap \bigcap_{j=1}^{\ell} H_j$. The optimal value of the problem is $f^* = \inf f(x)$ s.t. $g_i(x) \leq 0, \qquad \text{for } i \in [m], h_j(x) = 0, \qquad \text{for } j \in [\ell]$.

A crucial observation we want to leverage in this chapter is that one can use the following indicator functions $I_-,\ I_0 :\ \mathbb{R} \to \mathbb{R} \cup \{\infty\}$ defined as:

$$I_-(x) = \begin{cases} 0 & \text{for } x \leq 0, \\ \infty & \text{for } x > 0, \end{cases}$$

and

$$I_0(x) = \begin{cases} 0 & \text{for } x = 0, \\ \infty & \text{for } x \neq 0, \end{cases}$$

to rewrite a generic mathematical program into an unconstrained program of the form:

$$\min f(x) + \sum_{i=1}^{m} I_-\left(g_i(x)\right) + \sum_{j=1}^{\ell} I_0\left(h_j(x)\right).$$

It is easy to see that the reformulation is equivalent to the original program. Indeed, for every point $x \in \mathbb{R}^n$ that satisfies all the constraints of the original program, we have $\sum_{i=1}^{m} I_-\left(g_i(x)\right) + \sum_{j=1}^{\ell} I_0\left(h_j(x)\right) = 0$ thus, the value of the objective function remains unchanged. In turn, for a point $x \in \mathbb{R}^n$ that violates at least one of the original constraints, the corresponding indicator function penalizes the objective function with an infinite additional cost.

Note that we did not require the functions g_i for $i \in [m]$ to be convex on purpose as the above transformation works for any type of function. However, after the transformation, the resulting objective function is also not a convex function. Even if the original mathematical program would be a convex program, the resulting program would not be convex after the transformation. This simply comes from the definition of the indicator function. In the following, we approximate the indicator functions such that the composition of the approximation functions and the constraints yields a lower bound on the composition of the indicator functions and the constraints. For the case when the original mathematical program is a convex program, this will serve as the main building block to establish duality for convex programs.

There are many possibilities to lower bound the composition of the indicator function and the constraints. A simple, and as we learn later, very powerful one is to lower bound the indicator function alone with a linear function of the form

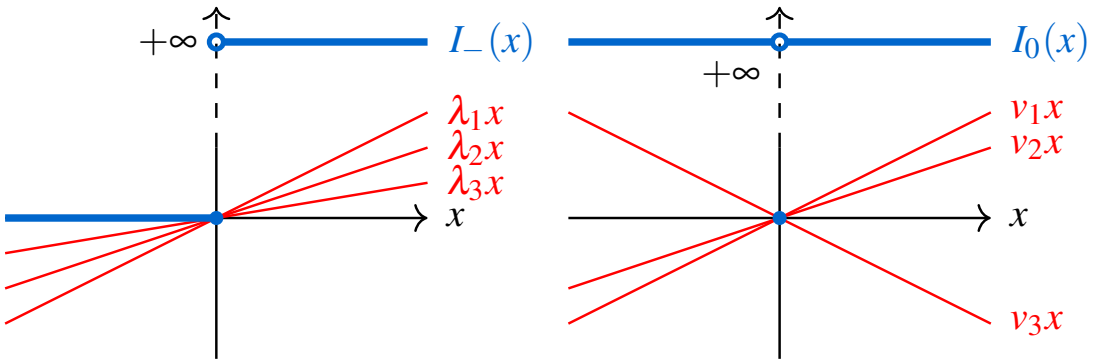
$$\lambda_i x_i \leq I_-(x_i)$$

and

$$v_i x_i \leq I_0(x_i)$$

for $i \in [n]$, $\lambda \in \mathbb{R}^n_{\geq 0}$ and $v \in \mathbb{R}^n$, respectively. Visualization of sample linear lower bounds on the indicator functions can be seen in the Example 3.1.1 below.

■ **Example 3.1.1** Linear lower bound on the indicator functions $I_-(x)$ on the left and on $I_0(x)$ on the right, for $n = 1$. Vectors $\lambda_1, \lambda_2, \lambda_3$ are in $\mathbb{R}_{\geq 0}$ and vectors v_1, v_2, v_3 are in $\mathbb{R}$.



■

This lead to the definition of a Lagrangian.

拉格朗日对偶方程就是求拉格朗日的inf

Definition 3.1.2 — Lagrangian. For a mathematical program the function $L\colon \mathbb{P}\times\mathbb{R}^m\times\mathbb{R}^\ell\to\mathbb{R}$ defined as

$$L(x,\lambda,\nu) := f(x)+\sum_{i=1}^{m}\lambda_i g_i(x)+\sum_{j=1}^{\ell}\nu_j h_j(x)$$

is called the *Lagrangian*. Moreover, the vectors $\lambda\in\mathbb{R}^m$ and $\nu\in\mathbb{R}^\ell$ are called *Lagrangian multipliers*.

As mentioned above, for each $\lambda\in\mathbb{R}^m_{\geq 0}$ and $\nu\in\mathbb{R}^\ell$ the corresponding Lagrangian at point $x\in\mathbb{R}^n$ yields a lower bound on the value of the mathematical program at point x. We start formalizing this concept with the following definition.

Definition 3.1.3 — Lagrange dual function. For a mathematical program the function $\widehat{L}\colon \mathbb{R}^m\times\mathbb{R}^\ell\to\mathbb{R}\cup\{-\infty\}$ defined as

$$\widehat{L}(\lambda,\nu) := \inf_{x\in\mathbb{P}} L(x,\lambda,\nu)$$

is called the *Lagrange dual function*. If the Lagrangian is unbounded below in x, we set $\widehat{L}=-\infty$.

The Lagrange dual function has a fundamental property. Namely, it is a concave function, even if the underlying mathematical program is not a convex program.

Proposition 3.1.4 The Lagrange dual function is a concave function.

针对单个点进行操作

Proof. By Proposition 1.1.11 the pointwise maximum of convex functions is convex. This property extends to taking the supremum of an infinite set. Thus, the pointwise infimum of a set of concave functions is concave.
Now note that the Lagrange dual function is a pointwise infimum of affine functions in variables λ_i for $i\in[m]$ and ν_j for $j\in[\ell]$. Since an affine function is a concave function the proposition holds. ■

We are ready to present the definition of the dual feasible region and a dual program. The names indicate the connection between the concepts discussed so far and the duality theory we want to establish in this chapter.

Definition 3.1.5 — dual feasible region. The set

$$\mathscr{D} := \{(\lambda,\nu)\in\mathbb{R}^m_{\geq 0}\times\mathbb{R}^\ell \mid \widehat{L}(\lambda,\nu)>-\infty\}$$

is called the *dual feasible region*.

Definition 3.1.6 — Lagrange dual program. The *Lagrange dual program* has the form

$$\begin{aligned}\widehat{f}^* := \max\ & \widehat{L}(\lambda,\nu),\\ & \lambda\in\mathbb{R}^m_{\geq 0},\\ & \nu\in\mathbb{R}^n.\end{aligned}$$

In many cases it is possible to derive an explicit formulation for the dual program. In the following, we show an example of how the construction can be applied to get an explicit formulation of the dual program for a linear program in standard form.

■ **Example 3.1.7** Consider a linear program in standard form as in Definition 1.4.1

$$\begin{aligned} \min\ & c^\top x \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. Let $\lambda \in \mathbb{R}^n$ and $\nu \in \mathbb{R}^m$ be the Lagrangian multipliers. We get the Lagrangian of the form

$$L(x,\lambda,\nu) = c^\top x - \sum_{i=0}^{n} \lambda_i x_i + \nu^\top (Ax-b) = -b^\top \nu + \Big(c + A^\top \nu - \lambda\Big)^\top x.$$

Note that a minus sign in front of the summation comes from the fact that we have to flip inequalities, i.e., write $-x \leq 0$ to match the definition of the Lagrangian. Hence, the dual function takes the form

$$\widehat{L}(\lambda,\nu) = \inf_x L(x,\lambda,\nu) = -b^\top \nu + \inf_x \Big(c + A^\top \nu - \lambda\Big)^\top x.$$

The linear function $(c + A^\top \nu - \lambda)^\top x$ is homogeneous, thus bounded below only if it is identically zero, which gives the following explicit form of the dual function

$$\widehat{L}(\lambda,\nu) = \begin{cases} -b^\top \nu & \text{, for } c + A^\top \nu - \lambda = 0, \\ -\infty & \text{, otherwise.} \end{cases}$$

The dual problem can be equivalently written as

$$\begin{aligned} \max\ & -b^\top \nu \\ & c + A^\top \nu - \lambda = 0 \\ & \lambda_i \geq 0 \qquad \text{for } i \in [n] \end{aligned}$$

which can be equivalently expressed as

$$\begin{aligned} \max\ & -b^\top \nu \\ & c + A^\top \nu \geq 0 \end{aligned}$$

which coincides with the formulation presented in Definition 1.4.5, after flipping the sign of the decision variables. ■

We finalize the section with several examples showing that the formulation of the dual problem heavily depends on the formulation of the primal problem. This applies even for the case when we modify the explicit description of the primal problem, without changing the problem. An example transformation can be, e.g., replacing the objective function with an increasing function of the objective function or changing the description of the feasibility set, by adding redundant variables or constraints. One of such examples is shown below.

■ **Example 3.1.8** Consider an unconstrained problem of the form

$$\min f(Ax+b).$$

Since the problem is unconstrained the Lagrangian is $L(x) = f(Ax+b)$ and dual function is $\widehat{L} = \inf_x f(Ax+b)$, thus the dual program is just a constant f^* which is an optimal value for the primal problem.

A simple transformation of the problem, by adding a redundant variable, leads to the following program

$$\begin{aligned} &\min f(y),\\ &\quad Ax+b=y. \end{aligned}$$

Now the the Lagrangian is

$$L(x,y,\nu) = f(y)+\nu^\top (Ax+b-y)$$

and dual function is

$$\widehat{L}(\nu) = b^\top \nu + \inf_x \left(\nu^\top Ax\right) + \inf_y \left(f(y)-\nu^\top y\right),$$

which is $-\infty$ unless $A^\top \nu = 0$ thus we get

$$\widehat{L}(\nu) = b^\top \nu - f^*(\nu),$$

where $f^*(y) \coloneqq \sup_y \left(\nu^\top y - f(y)\right)$ is called a *conjugate function* of f. The dual program takes the form

$$\begin{aligned} &\max b^\top \nu - f^*(\nu),\\ &\quad A^\top \nu = 0, \end{aligned}$$

which is potentially more useful than the original dual problem. ∎

3.2 Weak duality

We are ready to formalize the statement that the Lagrange dual function yields a lower bound on the optimal solution of the underlying mathematical program. As discussed at the beginning of this chapter, our goal is to construct a family of lower bounds on the optimal value of the original mathematical program. This family is given by the family of Lagrange dual functions, each of which corresponds to a pair of vectors $(\lambda,\nu) \in \mathbb{R}^m_{\geq 0} \times \mathbb{R}^\ell$. The Lagrange dual program aims to choose the one that maximizes the optimal value, i.e., minimizes the gap from the optimal value of the original problem. This is formalized in the following theorem that proves weak duality for mathematical programs.

Let f^* be an optimal value of the mathematical program and $\widehat{f}^*$ be an optimal value of the corresponding Lagrange dual program.

Theorem 3.2.1 — weak duality. For every $(\lambda,\nu) \in \mathscr{D}$ it holds that

$$\widehat{L}(\lambda,\nu) \leq f^*.$$

In particular, it holds that

$$\widehat{f}^* \leq f^*.$$

Proof. Let $\tilde{x}$ be a feasible point for the mathematical program, then $g_i(\tilde{x}) \leq 0$ for all $i \in [m]$ and $h_j(\tilde{x}) = 0$ for all $j \in [\ell]$. Thus we have

$$\sum_{i=1}^m \lambda_i g_i(\tilde{x}) + \sum_{j=1}^\ell \nu_j h_j(\tilde{x}) \leq 0.$$

Hence

$$\widehat{L}(\lambda, \nu) = \inf_{x \in \mathbb{P}} L(x, \lambda, \nu) \leq L(\tilde{x}, \lambda, \nu) \leq f(\tilde{x}).$$

In particular, since the above inequality holds for every $(\lambda, \nu) \in \mathscr{D}$ we get that $\widehat{f}^* \leq f(\tilde{x})$ and as it further holds for every feasible $\tilde{x}$ we conclude $\widehat{f}^* \leq f^*$. ■

Dual programs, by weak duality, serve as a great tool to derive bounds on the optimal value of the optimization problem. Especially, for the case when we deal with a non-convex programs, the bound obtained by the dual program can be very valuable, since, by Proposition 3.1.4, the dual program is always convex. One such example in presented below.

■ **Example 3.2.2** Let W be a real symmetric matrix. The *Two-way Partitioning (TWP)* problem is defined as:

$$\begin{aligned} \min \ & x^\top W x, \\ & x_i^2 = 1 \qquad \text{for all } i \in [n]. \end{aligned}$$

A natural interpretation of the TWP problem is given from the graph theory perspective. Let K_n be a complete graph with n vertices. And let the W_{ij} for $i, j \in [n]$ be the weight of the edge (i, j). The TWP problem consists in partitioning the set of vertices into subsets S and $V \setminus S$ such that the sum over all the weights of edges being on the same side of the partition minus the sum over all the weights of edges having two end-points on different sides of the partition is minimized. A special case of the problem is, e.g., the Max-Cut problem discussed in detail later in this book. The problem is non-convex, and it is hard to solve on modern computers for n bigger than the value of 40-50.

However, we can use the dual program to derive a lower bound on the value of the optimal solution for the TWP problem. We start with providing an explicit formulation of the dual program.

The Lagrangian for the TWP problem takes the following form

$$\begin{aligned} L(x, \nu) &= x^\top W x + \sum_{i=1}^{n} \nu_i \left(x_i^2 - 1\right) \\ &= x^\top \left(W + \operatorname{diag}(\nu)\right) x - \mathbb{1}^\top \nu. \end{aligned}$$

Note that the infimum of a quadratic form $x^\top \left(W + \operatorname{diag}(\nu)\right) x$ is either zero or minus infinity. Thus we get the following dual program

$$\begin{aligned} \widehat{L}(\nu) &= \inf_x \ x^\top \left(W + \operatorname{diag}(\nu)\right) x - \mathbb{1}^\top \nu \\ &= \begin{cases} -\mathbb{1}^\top \nu & \text{for } \lambda_{\min}\left(W + \operatorname{diag}(\nu)\right) \geq 0, \\ -\infty & \text{otherwise.} \end{cases} \end{aligned}$$

This leads to the following program

$$\begin{aligned} \max \ & -\mathbb{1}^\top \nu, \\ & -\lambda_{\min}\left(W + \operatorname{diag}(\nu)\right) \leq 0, \end{aligned}$$

which is a convex program. Indeed, the objective is a linear function and the constraint inequality involves convex functions. This can be seen either by applying a similar technique as in the Exercise 1.5 or directly by noting that for a symmetric matrix M the minimum eigenvalue of M is defined as $\lambda_{\min}(M) = \min_{\|v\|=1} v^\top M v$ and taking minimum over the family of liner functions (that are both convex and concave) is concave. The last argument follows since, by Proposition 1.1.11

the pointwise maximum of convex functions is convex. Thus, the pointwise minimum of a set of concave functions is concave.

The dual program is a convex relaxation of the original program and provides a lower bound on the optimal value of the original program. A feasible solution for the dual program is $v = -\lambda_{\min}(W)\mathbb{1}$, since

$$\lambda_{\min}(W - \lambda_{\min}(W)I) \geq 0,$$

which yields a lower bound of n $n \cdot \lambda_{\min}(W)$.

■

3.3 Strong duality

As mentioned above, a dual program constitutes a lower bound on the optimal value of the mathematical program. Ideally, we would like this lower bound to match the optimal value. However, this is not always the case. This leads to the definition of the duality gap.

Definition 3.3.1 — strong duality. For a primal and a dual mathematical program with optimal values f^* and $\widehat{f}^*$ respectively, the difference

$$f^* - \widehat{f}^* \geq 0$$

is called the *duality gap* of the mathematical program if either value is finite. Moreover, if the duality gap is zero ($f^* = \widehat{f}^*$ if both are infinite), we say that *strong duality* holds for the mathematical program.

Clearly, strong duality is a very desirable property. However, strong duality does not always hold. We are interested in investigating the benefits of strong duality and the conditions under which strong duality holds. We start with the first topic.

One of the very interesting and widely used properties of strong duality is the complementary slackness condition.

Proposition 3.3.2 — Complementary slackness. Let x^* and (λ^*, v^*) be optimal primal and optimal dual solutions respectively, such that $f^* = \widehat{f}^*$ (strong duality holds). Then for every $i \in [m]$ at most one of $\lambda_i^* > 0$ and $g_i(x_i^*) < 0$ can hold.

Proof. Since x^* and (λ^*, v^*) are primal and dual optimal we have

$$\begin{aligned} f(x^*) &= \widehat{L}(\lambda^*, v^*) \\ &= \inf_x \left(f(x) + \sum_{i=0}^{m} \lambda_i^* g_i(x) + \sum_{j=1}^{\ell} v_j^* h_j(x) \right) \\ &\leq f(x^*) + \sum_{i=0}^{m} \lambda_i^* g_i(x^*) + \sum_{j=1}^{\ell} v_j^* h_j(x^*) \\ &\leq f(x^*), \end{aligned}$$

where the first equality comes from strong duality, the second comes directly from the Definition 3.1.3 of the Lagrange dual function. The first inequality holds since x^* is a feasible solution thus the infimum is smaller than the evaluation of the Lagrangian at x^* and the last inequality applies since $\lambda_i^* g_i(x^*) \leq 0$ for all $i \in [m]$ and $v_j^* h_j(x^*) = 0$ for all $j \in [\ell]$.

The cornerstone observation is that the inequalities above hold with equality; thus, in particular, $\sum_{i=0}^{m} \lambda_i^* g_i(x^*) = 0$ and since every term in the summation is nonpositive, we have

$$\lambda_i^* g_i(x^*) = 0 \qquad \text{for all } i \in [m].$$

This implies that for each $i \in [m]$ at most one of the factors λ_i^* and $g_i(x^*)$ can be nonzero. ■

A practical implication of the complementary slackness is that for a primal and dual pair of optimal solutions, if a dual variable takes a nonzero value, then the corresponding primal constraint has to be active when evaluated at a primal optimal solution, i.e.,

$$\lambda_i^* > 0 \Rightarrow g_i(x^*) = 0.$$

Conversely, if a primal constraint evaluated at an optimal solution has a slack, then the corresponding dual variable is zero, i.e.,

$$g_i(x^*) < 0 \Rightarrow \lambda_i^* = 0.$$

3.4 KKT conditions

In this section, we make a step towards understanding the necessary conditions under which strong duality holds. For this to happen we make the assumption that the functions f, g_i for $i \in [m]$ and h_j for $j \in [\ell]$ are differentiable. This assumption is not very restrictive in practical applications.

Before we start, let us elaborate on the potential benefit of the differentiability that we have assumed. Let x^* and (λ^*, ν^*) be primal and dual optimal solutions for a problem with zero duality gap. Since x^* minimizes the Lagrangian $L(x, \lambda^*, \nu^*)$ over x, it follows that its gradient must vanish at x^*, i.e.,

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) + \sum_{j=1}^{\ell} \nu_j^* \nabla h_j(x^*) = 0.$$

Let us formalize this concept. We start with the following definition.

Definition 3.4.1 A point $\tilde{x}, \tilde{\lambda}, \tilde{\nu} \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^\ell$ is called a *KKT-point* if it satisfies the following *KKT-conditions*:

$$g_i(\tilde{x}) \leq 0, \quad \text{for all } i \in [m], \tag{KKT-1}$$
$$h_j(\tilde{x}) = 0, \quad \text{for all } j \in [\ell], \tag{KKT-2}$$
$$\tilde{\lambda}_i \geq 0, \quad \text{for all } i \in [m], \tag{KKT-3}$$
$$\tilde{\lambda}_i g_i(\tilde{x}) = 0, \quad \text{for all } i \in [m], \tag{KKT-4}$$
$$\nabla f(\tilde{x}) + \sum_{i=1}^{m} \tilde{\lambda}_i \nabla g_i(\tilde{x}) + \sum_{j=1}^{\ell} \tilde{\nu}_j \nabla h_j(\tilde{x}) = 0. \tag{KKT-5}$$

the gradient to vanish

Proposition 3.4.2 For any mathematical program with differentiable functions f, g_i, h_j for $i \in [m]$ and $j \in [\ell]$ in which strong duality holds, any pair $x^*, (\lambda^*, \nu^*)$ of primal and dual optimal solutions form a KKT-point.

Proof. Let $x^*, (\lambda^*, \nu^*)$ be a pair of primal and dual optimal solutions for a mathematical optimization problem with no duality gap. We show that the point x^*, λ^*, ν^* is a KKT-point. Indeed, since x^* is a primal optimal solution, the first and second condition follow from the feasibility of x^*. The third condition follows from the feasibility of the dual optimal solution λ^*, ν^* and the fourth condition follows from complementary slackness for an optimal pair of solutions $x^*, (\lambda^*, \nu^*)$ and the strong duality. Finally, the last condition is satisfied since x^* minimizes the Lagrangian $L(x, \lambda^*, \nu^*)$ over x; thus, its gradient must vanish at x^*. ■

So far, we did not assume any convexity of the objective function and the constraints. Assuming the convexity of the objective function and the constraints, a converse of the above proposition also holds. More precisely.

try to find a KKT point that is the optimal dual and optimal primal

Proposition 3.4.3 For any convex program with differentiable functions f, g_i, h_j for $i \in [m]$ and $j \in [\ell]$ if x^*, λ^*, v^* is a KKT-point then x^* and λ^*, v^* are primal and dual optimal solutions and strong duality holds.

Proof. First we observe that (KKT-1) and (KKT-2) imply that x^* is feasible for the primal program. By (KKT-3) the Lagrangian $L(x, \lambda^*, v^*)$ is a convex function in x. Thus any zero of its gradient is a global minimizer. The final condition (KKT-5) now states that the gradient vanishes at x^*. Hence $\widehat{L}(\lambda^*, v^*) = L(x^*, \lambda^*, v^*)$ by definition of the Lagrange dual function. Inserting the definition of the Lagrangian we conclude:

$$\begin{aligned} L(x^*, \lambda^*, v^*) &= f(x^*) + \sum_{i=1}^{m} \lambda^* g_i(x^*) + \sum_{j=1}^{p} v_j h_j(x^*) \\ &\overset{(KKT-2)}{=} f(x^*) + \sum_{i=1}^{m} \lambda^* g_i(x^*) \\ &\overset{(KKT-4)}{=} f(x^*). \end{aligned}$$

Thus x^* as a primal and λ^*, v^* as a dual solution have zero duality gap. For any primal solution x we have $f(x) \geq \widehat{L}(\lambda^*, v^*) = f(x^*)$ and conversely for any dual solution λ, v we have $\widehat{L}(\lambda, v) \leq f(x^*) = \widehat{L}(\lambda^*, v^*)$. This proves the claimed optimality of x^* and λ^*, v^*. ■

As we have seen the KKT-conditions can serve as a method to decide whether the duality gap of a convex program is zero, i.e., whether strong duality holds. In some cases, it is even possible to solve KKT-conditions analytically, thus getting a KKT-point and solving the optimization problem (and its dual problem at the same time). Moreover, many algorithms for solving convex optimization problem can be seen as methods to find a KKT-point.

However, it is important to note that the discussion about the KKT-conditions does not exhaust the topic about strong duality for convex programs. The main reason is that in some cases there might be no solution for the KKT-conditions, i.e., no KKT-point although strong duality still holds. Such a situation is visualized in the following example.

■ **Example 3.4.4** Consider the following mathematical program

$$\begin{aligned} &\min x \\ &\quad x^2 \leq 0 \end{aligned}$$

The program is convex since the objective is an affine function, and the constraint is a convex function. The unique optimal solution for the program is 0 for $x = 0$, which is the only feasible solution for the program.
The Lagrangian of the program is

$$L(x, \lambda) = x + \lambda x^2$$

and the Lagrange dual function is given by

$$\widehat{L}(\lambda) = \inf_{x \in \mathbb{R}^n} L(x, \lambda) = \begin{cases} -\infty & \text{for } \lambda \leq 0, \\ -\frac{1}{4\lambda} & \text{for } \lambda > 0. \end{cases}$$

This implies that the dual optimal value is also 0; thus, the duality gap is zero, i.e., strong duality holds. Note, however, that the dual optimal solution is never attained, i.e., it is attained as $\lambda \to \infty$.

Now we analize the KKT-conditions. We look for a solution of the following system

$$\begin{aligned} 1+2\lambda x &= 0 \\ x^2 &\leq 0 \\ \lambda &\geq 0 \\ \lambda x^2 &= 0. \end{aligned}$$

The second condition implies that $x = 0$ which for any $\lambda \geq 0$, from the third condition, satisfies the fourth equality; however, it never satisfies the first equality. Thus, there is no KKT-point for this program. ∎

The Example 3.4.4 highlights several crucial observations. First, by Proposition 3.4.2, for differentiable functions f, g_i, h_j for $i \in [m]$ and $j \in [\ell]$ any pair of primal and dual optimal solutions $x^*, (\lambda^*, \nu^*)$ forms a KKT-point. Note however, that in Example 3.4.4 the dual optimum is never attained, which is the main obstacle that implies non existence of a KKT-point in this case. Second, as pointed out in the discussion above, solving KKT-conditions serves as a very useful technique for solving convex optimization problems, that is leveraged in many algorithms designed for this purpose. The correctness of such algorithm follows by Proposition 3.4.3 which says that for convex differentiable functions f, g_i, h_j for $i \in [m]$ and $j \in [\ell]$, any KKT-point x^*, λ^*, ν^* forms a primal optimal x^* and dual optimal λ^*, ν^* solutions with zero duality gap. However, Example 3.4.4 shows that this method might fail to find optimal solutions of the convex program, even if strong duality holds, since a KKT-point might not exist.

3.5 Strong duality — Slater's condition

There are many other conditions that certify strong duality for convex programs. The family of such conditions is called *constraint qualifications*. A widely used example of constraint qualification is Slater's condition. Let $\mathscr{P}$ be the feasibility set defined as $\mathscr{P} := \{x \in \mathbb{R}^n : \ g_i(x) \leq 0, \text{ for } i \in [m],\ h_j(x) = 0 \text{ for } j \in [\ell]\}$.

Theorem 3.5.1 — Slater's condition. For any convex program if *Slater's conditions* is satisfied, i.e., there exists a point $x^* \in \text{relint}(\mathscr{P})$, i.e., $g_i(x^*) < 0$ and $h_j(x^*) = 0$, then strong duality holds and the dual optimum value is attained.

We provide the proof of Slater's condition in the next section. There we give a new geometric perspective on the duality that will lead to the proof of Slater's condition. Before we do that, we discuss the statement of Slater's condition a bit deeper.

In the case when the first k constraints $g_1, \ldots, g_k$ are affine functions, a relaxed condition is sufficient for strong duality to hold. Namely, in such a case it is enough that there exists a point $x \in \mathbb{R}^n$ such that $g_i(x) < 0$ for $i \in \{k+1, \ldots, m\}$ and $g_i(x) = h_j(x) = 0$ for $i \in [k]$ and $j \in [\ell]$. As an immediate implication of the relaxed Slater's condition we get that strong duality always hold for linear programs. Second, note that in Example 3.4.4 a point x^* fulfilling Slater's condition would need to have $(x^*)^2 < 0$ which is impossible. In that case, Slater's condition thus does not hold; however, strong duality still holds. More importantly, the dual optimal value is not attained in that case. In folklore communication, sometimes the second part of Slater's condition is omitted, mainly focusing on the strong duality implication of Slater's condition. Example 3.4.4 shows that the second implication, the attainability of the dual optimal solution, is equally important to remember.

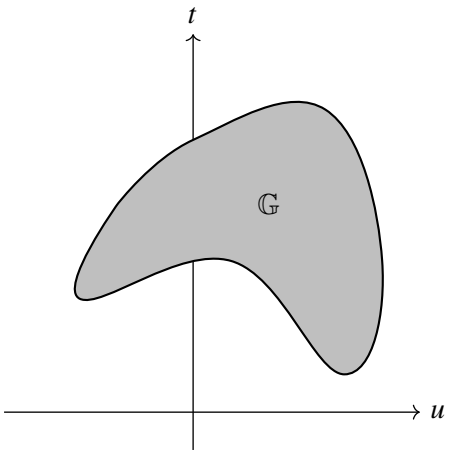


Figure 3.1: Example of the geometric duality interpretation $\mathbb{G}$ for a Mathematical Optimization Problem.

3.6 Geometric interpretation

In this section, we present another perspective on duality based on a geometric interpretation of the concepts discussed so far. Let us define the set of values taken by the constraints and the objective function as follows

$$\mathbb{G} := \left\{ (g_1(x), \ldots, g_m(x), h_1(x), \ldots, h_\ell(x), f(x)) \in \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R} : \; x \in \mathbb{P} \right\}.$$

■ **Example 3.6.1** To illustrate this concept, we look at a mathematical problem with $m = 1$ and $\ell = 0$. Then a possible region $\mathbb{G} \subseteq \mathbb{R}^2$ can be seen in Figure 3.1. We will keep using this region to illustrate the following concepts. Already now one may ask if strong duality holds in the depicted example. ■

Given the definition of the set $\mathbb{G}$, we can express the optimal value f^* and Lagrange dual functions as:

$$f^* = \inf \{ t : \; (u, w, t) \in \mathbb{G}, \; u \leq 0, \; w = 0 \}$$

$$\widehat{L}(\lambda, \nu) = \inf \{ (\lambda, \nu, 1)^\top (u, w, t) : \; (u, w, t) \in \mathbb{G} \},$$

where

$$(\lambda, \nu, 1)^\top (u, w, t) = \sum_{i=1}^{m} \lambda_i u_i + \sum_{j=1}^{\ell} \nu_i w_i + t$$

is the evaluation of the Lagrange dual function at the point (λ, ν). Note that, the formula above corresponds to the Lagrangian, where u_i, w_j and t are the evaluation of $g_i(x), h_j(x)$ and $f(x)$ for some $x \in \mathbb{P}$. We make the following two observations. First, if $\widehat{L}(\lambda, \nu)$ is finite then we have for all $(u, w, t) \in \mathbb{G}$

$$(\lambda, \nu, 1)^\top (u, w, t) \geq \widehat{L}(\lambda, \nu).$$

which defines a *nonvertical supporting hyperplane* of the form $(u', w', t') \in H \subseteq \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}$, iff

$$(\lambda, \nu, 1)^\top (u', w', t') = \widehat{L}(\lambda, \nu)$$

■ **Example 3.6.2** Keeping the region $\mathbb{G}$ from Example 3.6.1, the nonvertical supporting hyperplanes for two different values of λ can be seen in Figure 3.2. They are attained by minimizing $\lambda u + 1t$ over $(u, t) \in \mathbb{G}$ which gives $\widehat{L}(\lambda)$ as discussed before. Note that indeed H is a supporting hyperplane as well as nonvertical. ■

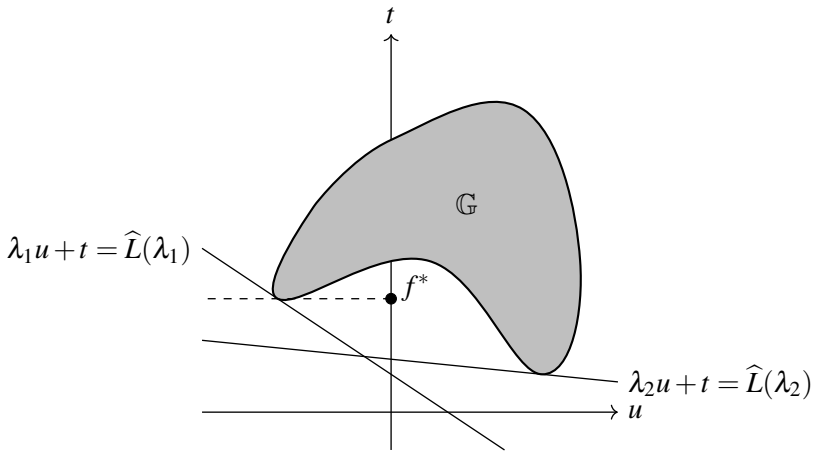


Figure 3.2: Example of two different nonvertical supporting hyperplanes for $\mathbb{G}$. The slopes are given by $-\lambda_i$ (this can be seen be rewriting $t = \widehat{L}(\lambda_i) - \lambda_i u$) for $i = 1, 2$.

The second observation is that if $\lambda \geq 0$, $u \leq 0$ and $w = 0$, then

$$t \geq (\lambda, \nu, 1)^\top (u, w, t)\,.$$

Using the geometric interpretation we get a straightforward alternative for the proof of weak duality for mathematical programs.

Alternative proof of Theorem 3.2.1
For every (λ, ν) it holds that

$$\begin{aligned}\widehat{L}(\lambda, \nu) &= \inf\{(\lambda, \nu, 1)^\top (u, w, t) :\ (u, w, t) \in \mathbb{G}\} \\ &\leq \inf\{(\lambda, \nu, 1)^\top (u, w, t) :\ (u, w, t) \in \mathbb{G},\ u \leq 0, w = 0\} \\ &\leq \inf\{t :\ (u, w, t) \in \mathbb{G},\ u \leq 0, w = 0\} = f^*,\end{aligned}$$

where the last inequality follows from the second observation discussed above. Since, the above inequality holds for every $(\lambda, \nu) \in \mathscr{D}$ we get that $\widehat{f}^* \leq f^*$. ■

■ **Example 3.6.3** In order to visualize the duality gap, we can plug $u = 0$ into $\lambda u + t = \widehat{L}(\lambda)$ and get that $\widehat{L}(\lambda)$ is given by the intersection of H with the t-axis. With the region from Example 3.6.1, this can be seen in Figure 3.3. Therefore we can finally answer the question from Example 3.6.1 as we now see that the duality gap is nonzero even for the optimum dual value λ^*. ■

The reasoning above can be further transformed using a concept related to the construction of an epigraph, which brings us closer to understanding why strong duality for convex programs is very natural. It will also serve as the main building block to prove strong duality under Slater's condition. More precisely, we define a set $\mathbb{A} \subseteq \mathbb{R}^m \times \mathbb{R}^\ell \times \mathbb{R}$ using the Minkowski sum as

$$\mathbb{A} := \mathbb{G} \oplus \left(\mathbb{R}^m_{\geq 0} \times \{0\} \times \mathbb{R}_{\geq 0}\right).$$

■ **Example 3.6.4** The set $\mathbb{A}$ finally can be added to all the previous considerations. In Figure 3.4 one can see the way $\mathbb{A}$ is constructed as well as why in general only nonnegative λ values still define a supporting hyperplane. ■

Alternatively, the set can be defined as

$$\mathbb{A} := \{(u, w, t) :\ \exists x \in \mathbb{P} \text{ such that } g_i(x) \leq u_i \text{ for all } i \in [m],\ h_j(x) = w_i \text{ for all } j \in [\ell],\ f(x) \leq t\}.$$

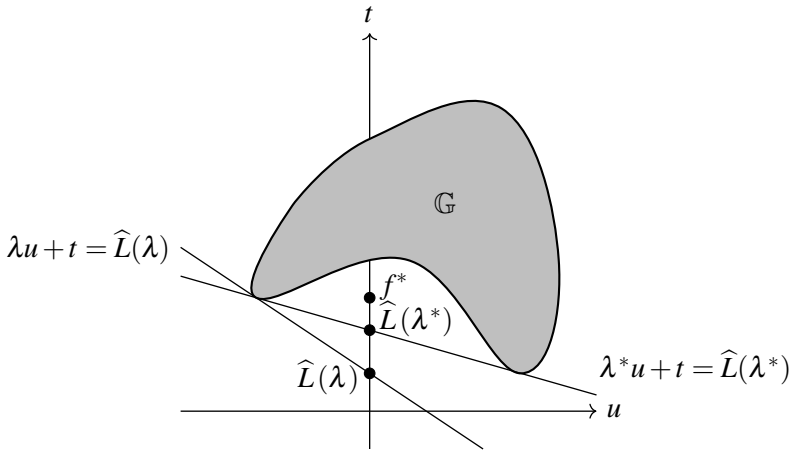


Figure 3.3: The nonvertical supporting hyperplanes of the dual optimal value λ^* and another nonoptimal λ for $\mathbb{G}$. The duality gap can be seen on the t-axis.

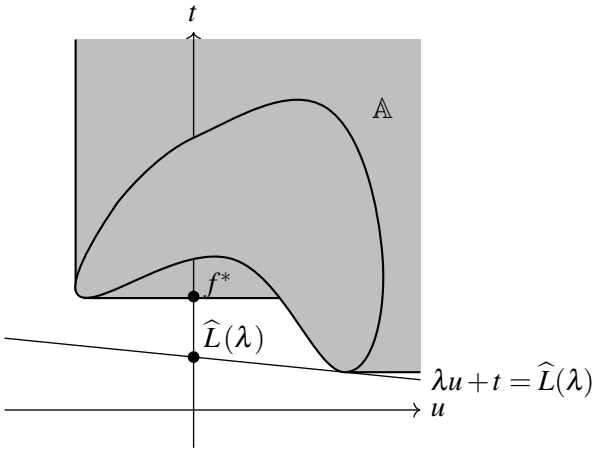


Figure 3.4: The nonvertical supporting hyperplanes of λ for the *epigraph* $\mathbb{A}$.

Then, the optimal value f^*, and Lagrange dual functions can be expressed respectively as:

$$f^* = \inf\{t : \ (0,0,t) \in \mathbb{A}\},$$

$$\widehat{L}(\lambda,\nu) = \inf\{(\lambda,\nu,1)^\top (u,w,t) : \ (u,w,t) \in \mathbb{A}\}.$$

Finally, we get another, very simple, alternative for the proof of weak duality for mathematical programs.

Alternative proof of Theorem 3.2.1
For every (λ,ν) it holds that

$$\widehat{L}(\lambda,\nu) \leq (\lambda,\nu,1)^\top (0,0,f^*) = f^*,$$

where the first equality follows from the fact that the point $(0,0,f^*) \in \mathbb{A}$. ■

We are now ready to prove Slater's condition.

Prove of Theorem 3.5.1
Recall the setting we assume. We are given a convex program

$$\begin{aligned} &\min f(x) \\ &\quad g_i(x) \leq 0, \qquad \text{for } i \in [m], \\ &\quad h_j(x) = 0, \qquad \text{for } j \in [\ell], \end{aligned}$$

and a point $x^* \in \operatorname{relint}(\mathscr{P})$ such that

$$\begin{aligned} &g_i(x^*) < 0, \qquad \text{for } i \in [m] \text{ and } g_i \text{ not affine}, \\ &h_j(x^*) = 0, \qquad \text{for } j \in [\ell]. \end{aligned}$$

We will only prove the case when $x^* \in \operatorname{int} \mathscr{P}$. In practice this is mostly the case, e. g. when f is differentiable. Extending to $x^* \in \operatorname{relint} \mathscr{P}$ uses the same ideas, but it would increase the notational overhead.

First we investigate the connection between dual feasible points and supporting hyperplanes of $\mathbb{A}$. As noticed above we have

$$\widehat{L}(\lambda,\nu) = \inf\{(\lambda,\nu,1)^\top (u,w,t) : \ (u,w,t) \in \mathbb{A}\},$$

thus whenever the infimum is finite and attained, $(\lambda,\nu,1)^\top (u,w,t) \geq \widehat{L}(\lambda,\nu)$ defines a supporting hyperplane at $\mathbb{A}$. Its intersection with the $(0,0,t)$ axis is exactly the value of the Lagrange dual function.

The other way around let $(\lambda,\nu,\mu)^\top (u,w,t) \geq L$ be a supporting hyperplane of $\mathbb{A}$. As $(0,0,f^*)$ is in $\mathbb{A}$, so is $(u,0,f^*)$ for all $u \geq 0$ by definition of $\mathbb{A}$. Thus

$$\lambda_i u_i + \mu f^* \geq L$$

for all non-negative u_i and all i. This is only possible when $\lambda_i \geq 0$, giving the first part of dual feasibility. With the same reasoning we get $\mu \geq 0$. If additionally $\mu > 0$ we can rescale:

$$\left(\lambda',\nu',1\right)^\top (u,w,t) \geq L'$$

where $\lambda' = \frac{\lambda}{\mu}, \nu' = \frac{\nu}{\mu}, L' = \frac{L}{\mu}$. As we started with a supporting hyperplane all points of $\mathbb{A}$ and in particular all points $(g(x), h(x), f(x))$ fullfill this equation. But this means that $\widehat{L}(\lambda', \mu') = L'$, as the left hand side is exactly the Langrangian.

The upshot is, that every supporting hyperplane with none vanishing final entry, which we call *non-vertical*, gives rise to a dual feasible point and its intersection with $(0,0,t)$ agrees with the value of the dual function.

Here non-verticality is important as else we could not normalize and hence would not get the connection to the Langrangian.

In order to relate this with strong duality we recall that $(0,0,f^*)$ is the infitisimal point in t where $(0,0,t) \in \mathbb{A}$. On the other hand each non-vertical hyperplane contains $(0,0,\widehat{L}(\lambda,\nu))$, hence strong duality holds if and only if $\mathbb{A}$ attains a non-vertical supporting hyperplane at $(0,0,f^*)$! In the following we will show the existence of such a hyperplane given a point x^* fulfils Slater's condition. For this assume that strong duality does not hold and additionally assume the problem has the minimal number of constraints under all counterexamples.

First we can apply the supporting hyperplane theorem, Theorem 1.2.8 to $\mathbb{A}$ at $(0,0,f^*)$ to get a supporting hyperplane $(\lambda,\nu,\mu)^\top (u,w,t) \geq \alpha$ with $(\lambda,\nu,\mu) \neq (0,0,0)$. This can be done since $\mathbb{A}$ is convex for convex optimization problems. As reasoned above we must have $\lambda \geq 0$ and $\mu \geq 0$. Furthermore we showed that strong duality holds when $\mu > 0$, so assume from now on $\mu = 0$.

We know, as the hyperplane is supporting at $(0,0,f^*)$ that

$$0 = \mu f^* = (\lambda,\nu,\mu)^\top (0,0,f^*) = \alpha,$$

and thus for all $x \in \mathscr{P}$:

$$(\lambda,\nu,0)^\top (g(x),h(x),f(x)) \geq 0. \tag{3.1}$$

Especially for x^* we have:

$$\sum_{i=1}^{m} \lambda_i g_i(x^*) + \sum_{j=1}^{\ell} \nu_j h_j(x^*) = \sum_{i=1}^{m} \lambda_i g_i(x^*) \geq 0.$$

As $\lambda \geq 0$ and $g_i(x^*) \leq 0$ we must have $\lambda_i = 0$ whenever $g_i(x^*) < 0$ so in particular for the non-affine g_i.

Plugging this into (3.1) yields

$$\sum_{j=1}^{\ell} \nu_j h_j(x) \geq 0. \tag{3.2}$$

Furthermore $\sum_{j=1}^{\ell} \nu_j h_j(x^*) = 0$. Now we use the second part of Slater's condition, namely that x lies in the interior of $\mathscr{D}$. Hence a small ball around x^* is completely in $\mathscr{P}$ and fulfils (3.2). But this is only possible if the affine function $\sum_{j=1}^{\ell} \nu_j h_j(x)$ is the zero function, as else it would decrease in some direction. Not all ν_j vanish as we started with a non-zero vector (λ,ν,μ). W.l.o.g. let $\nu_1 \neq 0$. Then

$$h_1 = \sum_{j=2}^{\ell} -\frac{\nu_j}{\nu_1} h_j(x).$$

But this means that the constraint of h_1 is redundant. Removing it yields an equivalent convex program with its dual being equivalent to the original dual, so for this program strong duality does not hold either. Hence the problem we started with was not a minimal counterexample, contradicting the assumptions. So indeed $\mu \neq 0$ and there are $\lambda \in \mathbb{R}^m_{\geq 0}, \nu \in \mathbb{R}^\ell$ with $\widehat{L}(\lambda,\nu) = f^*$. This concludes the proof that strong duality holds and the dual optimum is attained.

3.7 Theorems of alternatives

In this section we show how duality theory can be used to study feasibility of a system of inequalities and equalities. Consider the following system

$$\begin{aligned} g_i(x) &\leq 0 \quad \text{for } i \in [m], \\ h_j(x) &= 0 \quad \text{for } j \in [\ell]. \end{aligned} \tag{3.3}$$

Feasibility of this system can be decided by transforming it into a mathematical program with zero objective function. The obtained program and its dual have the following form

Primal:

$$\begin{aligned} f^* := \min \quad & 0 \\ & g_i(x) \leq 0, \qquad \text{for } i \in [m], \\ & h_j(x) = 0, \qquad \text{for } j \in [\ell] \end{aligned}$$

Dual:

$$\begin{aligned} \widehat{f}^* := \max \quad & \widehat{L}(\lambda, \nu) = \inf_{x \in \mathbb{P}} \left(\sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{\ell} \nu_j h_j(x) \right) \\ & \lambda \in \mathbb{R}^m_{\geq 0}, \\ & \nu \in \mathbb{R}^{\ell}. \end{aligned}$$

The only difference between the above program and the programs studied in this chapter is the objective function. For this reason, in this section, we use notations introduced earlier in this chapter.

Let us consider the following system of inequalities and equalities

$$\begin{aligned} \widehat{L}(\lambda, \nu) &> 0 \\ \lambda &\in \mathbb{R}^m_{\geq 0}, \\ \nu &\in \mathbb{R}^{\ell}. \end{aligned} \tag{3.4}$$

Note that, since $f(x) = 0$, the dual function is homogeneous in (λ, ν). That is, for any $\alpha > 0$, we have $\widehat{L}(\alpha \cdot \lambda, \alpha \cdot \nu) = \alpha \cdot \widehat{L}(\lambda, \nu)$. Thus we get that the primal and the dual optimal values take the following form

Primal:

$$f^* = \begin{cases} 0 & \text{iff (3.3) is feasible} \\ \infty & \text{iff (3.3) is infeasible} \end{cases}$$

Dual:

$$\widehat{f}^* = \begin{cases} 0 & \text{iff (3.4) is infeasible} \\ \infty & \text{iff (3.4) is feasible} \end{cases}$$

We introduce the following definition.

Definition 3.7.1 Two systems of inequalities and equalities are *weak alternatives* if at most one of the two is feasible.

We immediately get the following proposition.

Proposition 3.7.2 The two systems (3.3) and (3.4) are *weak alternatives*.

Proof. By weak duality we know that $\widehat{f}^* \leq f^*$. Thus if (3.3) is feasible we have $f^* = 0$ which, by weak duality, implies that $\widehat{f}^* = 0$ and the system (3.4) is infeasible. If (3.4) is feasible, we have $\widehat{f}^* = \infty$ which implies that $f^* = \infty$ and the system (3.3) is infeasible. ■

In a similar manner, the feasibility of a system of strict inequalities and equalities and its dual are weak alternatives. More precisely, consider the following two systems.

$$\begin{array}{lll} g_i(x) < 0 & \text{for } i \in [m] & \widehat{L}(\lambda, \nu) \geq 0 \\ h_j(x) = 0 & \text{for } j \in [\ell] & \lambda \in \mathbb{R}^m_{\geq 0}, \lambda \neq 0 \\ & & \nu \in \mathbb{R}^\ell. \end{array} \tag{3.5}$$

Proposition 3.7.3 The two systems in (3.5) are *weak alternatives*.

Proof. Assume x^* is a solution to the first system, i.e. $g_i(x^*) < 0, h_j(x^*) = 0$ then for any $\lambda \in \mathbb{R}^m_{\geq 0}, \nu \in \mathbb{R}^\ell$ surely

$$\sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^\ell \nu_j h_j(x) = \sum_{i=1}^m \underbrace{\lambda_i}_{\geq 0, \neq 0} \underbrace{g_i(x)}_{< 0} < 0.$$

So in particular $\widehat{L}(\lambda, \nu) < 0$ and thus the second system is infeasible. This proves the desired alternative statement. ■

Preposition 3.7.2 and Proposition 3.7.3 hold without any restriction on the type of constraints g_i and h_j. Note that the inequalities involved in the system corresponding to the dual are always convex. In the remainder of the section, we assume that the constraints involved in the system corresponding to the primal are also convex, that is, g_i for $i \in [m]$ are convex, and h_j for $j \in [\ell]$ are affine. In that case, we get even stronger theorem of alternatives.

Definition 3.7.4 Two systems of inequalities and equalities are *strong alternatives* if exactly one of the two is feasible.

This time we start with a system that involves strict inequalities first. Consider the following two systems

$$\begin{array}{lll} g_i(x) < 0 & \text{for } i \in [m] & \widehat{L}(\lambda, \nu) \geq 0 \\ Ax = b & \text{for } A \in \mathbb{R}^{\ell \times n} & \lambda \in \mathbb{R}^m_{\geq 0} \;\; \lambda \neq 0, \\ & & \nu \in \mathbb{R}^\ell. \end{array} \tag{3.6}$$

Proposition 3.7.5 For the systems in (3.6) if there exists a point $\tilde{x} \in \text{relint}(\mathscr{P})$ such that $A\tilde{x} = b$ then the two systems in (3.6) are *strong alternatives*.

Proof. The idea is to use convexity and Slater's condition. But applying Slater directly does not work as we can only apply it when the first system is feasible and even then it cannot handle the $\lambda \neq 0$ condition.

The trick is to first introduce a slack variable to the first system. So consider the following program instead:

$$\begin{array}{ll} \min s & \\ g_i(x) - s \leq 0, & \text{for } i \in [m], \\ Ax - b = 0, & \text{for } j \in [\ell], \end{array} \tag{3.7}$$

For it we can apply Slater as there is a $\tilde{x} \in relint(\mathscr{P})$ so $(\tilde{x}, s)$ satisfies Slater's condition for s sufficiently large, e.g. $s > \max_{i \in [m]} g_i(\tilde{x})$. So strong duality holds and the dual optimum is attained.

Let us calculate the dual then! We have for the Lagrange dual function of the new problem:

$$\widehat{L}'(\lambda, \nu) = \inf_{x \in \mathbb{P}, s \in \mathbb{R}} \left(s + \sum_{i=1}^m \lambda_i (g_i(x) - s) + \nu^\top (Ax - b) \right).$$

The sum above contains a term $(1-\sum_{i=1}^m \lambda_i)s$ so if $\sum_{i=1}^m \lambda_i \neq 1$ we could choose s arbitrarily big or small to show that the infimum is $-\infty$. Hence:

$$\widehat{L}'(\lambda,\nu) = \begin{cases} \widehat{L}(\lambda,\nu) & \text{if } \sum_{i=1}^m \lambda_i = 1, \\ -\infty & \text{else.} \end{cases}$$

Here $\widehat{L}(\lambda,\nu)$ denotes the Lagrange dual function of the original problem. So overall the dual problem of (3.7) is:

$$\begin{aligned} &\max \widehat{L}(\lambda,\nu) \\ &\quad \lambda \in \mathbb{R}^m_{\geq 0}, \ \sum_{i=1}^m \lambda_i = 1 \\ &\quad \nu \in \mathbb{R}^\ell. \end{aligned} \tag{3.8}$$

By Slater the optimal value f^* of (3.7) and $\widehat{f}^*$ of (3.8) agree. If both are negative, then the original program, the first one of (3.6) is feasible. Else $\widehat{f}^* \geq 0$ and this optimum is attained. Thus there are λ, ν with $L(\lambda,\nu) \geq 0$ and $\lambda \geq 0$, $\sum_{i=1}^m \lambda_i = 1$ in particular, $\lambda \neq 0$. So in this case the second program of (3.6) is feasible.

This shows that exactly one of the programs is feasible, showing the desired. ■

We finalize this section with a proposition about strong alternatives for systems involving convex nonstrict inequalities.

Consider the following two systems

$$\begin{aligned} g_i(x) &\leq 0 \qquad \text{for } i \in [m] \\ Ax &= b \qquad \text{for } A \in \mathbb{R}^{\ell \times n} \end{aligned} \qquad\qquad \begin{aligned} &\widehat{L}(\lambda,\nu) > 0 \\ &\lambda \in \mathbb{R}^m_{\geq 0} \ \ \lambda \neq 0, \\ &\nu \in \mathbb{R}^\ell. \end{aligned} \tag{3.9}$$

Proposition 3.7.6 For the systems in (3.9) if there exists a point $\tilde{x} \in relint(\mathscr{P})$ such that $A\tilde{x} = b$ and the optimal value f^* of the corresponding transformed problem (3.7) is attained then the two systems in (3.9) are *strong alternatives*.

Proof. We can do the analogous construction as in the proof of proposition 3.7.5. If $\widehat{f}^* > 0$ then the second system of (3.9) is feasible. Else $f^* = \widehat{f}^*$ and there is by assumption a pair (x,s) attaining the optimum. Then $s \leq 0$ and thus x is feasible for the first program in (3.9).

This combind with the weak alternative statement shown already in proposition 3.7.5 finishes the proof.

Observe that the only case for which we need the condition on f^* being attained is when $f^* = 0$. ■

3.8 Applications

3.8.1 Dual SVM

Recall the setting from Section 2.4, where we wanted to train a classifier to linearly separate two classes of points. We ended up with the following separation problem:

Input: A set of m, n-dimensional points $\{x_1, x_2, \ldots, x_m\} \in \mathbb{R}^n$, each associated with a $\{-1, 1\}$ label y_i.

Output: A hyperplane that separates points labeled with $+1$ from points labeled with -1 with a maximum possible margin.

To find this best separating hyperplane we formulated the problem as a convex optimization problem and finally we ended up with the following quadratic program:

$$\begin{aligned} \min \quad & \|w\|^2 \\ \text{s.t.} \quad & y_i(w^\top x_i + b) - 1 \geq 0 \qquad \text{for each } i \in \{1, \ldots, m\}. \end{aligned} \tag{P}$$

In this section we want to approach the same problem from the dual side. We will discuss the advantages of this approach at the end.

Let us begin with formulating the dual problem. For this we introduce a variable λ_i for each constraint. The Lagrangian of the primal problem (P) is then given by

$$L(w, b, \lambda) = \|w\|^2 - \sum_{i=1}^{m} \lambda_i \left(y_i(w^\top x_i + b) - 1 \right).$$

Here we have to subtract the functions of the inequality constraints as we consider non-negativity constraints instead of the usual non-positivity constrains. Next we have to determine the dual function

$$\widehat{L}(\lambda) = \inf_{w,b} \{L(w, b, \lambda)\} = \inf_{w,b} \left\{ \|w\|^2 - \sum_{i=1}^{m} \lambda_i \left(y_i(w^\top x_i + b) - 1 \right) \right\}.$$

We start by searching for local minima. For this we fix λ and set all partial derivatives to zero. This gives

$$\frac{d}{dw} L(w, b, \lambda) = 2w - \sum_{i=1}^{m} \lambda_i y_i x_i = 0 \tag{3.10}$$

and

$$\frac{d}{db} L(w, b, \lambda) = -\sum_{i=1}^{m} \lambda_i y_i = 0. \tag{3.11}$$

Equation (3.11) gives us that there can only be a local minimum if $\sum_{i=1}^{m} \lambda_i y_i = 0$. Indeed if this is not the case then varying b for fixed w and λ lets $L(w, b, \lambda)$ attain any real value. Thus in this case $\widehat{L}(\lambda) = -\infty$.

So assume $\sum_{i=1}^{m} \lambda_i y_i = 0$. Then $L(w, b, \lambda)$ is independent of b and for $\|w\|$ going to ∞ also $L(w, b, \lambda)$ goes to ∞ as the first summand grows quadratically while the second grows only linearly. To be more precise we have

$$\begin{aligned} L(w, b, \lambda) &= \|w\|^2 - \sum_{i=1}^{m} \lambda_i \left(y_i \left(w^\top x_i + b \right) - 1 \right) = \|w\|^2 - \sum_{i=1}^{m} \lambda_i \left(y_i(w^\top x_i) - 1 \right) \\ &\geq \|w\|^2 - \sum_{i=1}^{m} \lambda_i \left(y_i(\|w\| \, \|x\|) - 1 \right) = \|w\| \left(\|w\| - \sum_{i=1}^{m} \lambda_i y_i \, \|x\| \right) + \sum_{i=1}^{m} \lambda_i. \end{aligned}$$

From this we directly see that there has to be a local minimum and by (3.10) it has to be at the point $w = \frac{1}{2}\sum_{i=1}^m \lambda_i y_i x_i$. Plugging this into the Lagrangian gives the dual function:

$$\begin{aligned}\widehat{L}(\lambda) &= \left\| \frac{1}{2}\sum_{i=1}^m \lambda_i y_i x_i \right\|^2 - \sum_{i=1}^m \lambda_i \left(y_i \left(\left(\frac{1}{2}\sum_{j=1}^m \lambda_j y_j x_j \right)^\top x_i + b \right) - 1 \right) \\ &= \frac{1}{4}\sum_{i,j=1}^m \lambda_i \lambda_j y_i y_j \left(x_i^\top x_j \right) - \sum_{i,j=1}^m \frac{1}{2}\lambda_i \lambda_j y_i y_j \left(x_i^\top x_j \right) - b \underbrace{\sum_{i=1}^m \lambda_i y_i}_{=0} + \sum_{i=1}^m \lambda_i \\ &= \sum_{i=1}^m \lambda_i - \frac{1}{4}\sum_{i,j=1}^m \lambda_i \lambda_j y_i y_j \left(x_i^\top x_j \right).\end{aligned}$$

So putting all together the dual program is

$$\begin{aligned} \max \quad & \widehat{L}(\lambda) \\ \text{s.t.} \quad & \lambda_i \geq 0 \qquad \text{for each } i \in \{1,\dots,m\} \end{aligned}$$

where

$$\widehat{L}(\lambda) = \begin{cases} \sum_{i=1}^m \lambda_i - \frac{1}{4}\sum_{i,j=1}^m \lambda_i \lambda_j y_i y_j \left(x_i^\top x_j \right) & \text{if } \sum_{i=1}^m \lambda_i y_i = 0, \\ -\infty & \text{if } \sum_{i=1}^m \lambda_i y_i \neq 0. \end{cases}$$

Now as long as there are positive and negative y_i (and if they are not, the problem is not well defined and the classifier does not exist) the maximum value of g will be a real number which is surely attained with $\sum_{i=1}^m \lambda_i y_i = 0$. We can thus add this as a constraint and write $\widehat{L}(\lambda)$ explicitly giving the following final dual convex optimization problem:

$$\begin{aligned} \max \quad & \sum_{i=1}^m \lambda_i - \frac{1}{4}\sum_{i,j=1}^m \lambda_i \lambda_j y_i y_j \left(x_i^\top x_j \right) \\ \text{s.t.} \quad & \sum_{i=1}^m \lambda_i y_i = 0, \\ & \lambda_i \geq 0 \qquad \text{for each } i \in \{1,\dots,m\}. \end{aligned} \tag{D}$$

(3.12)

Now there is one problem remaining: Once we have found the optimal λ_i how do we classify a new point z? First recall that by (3.10) we have $w = \frac{1}{2}\sum_{i=1}^m \lambda_i y_i x_i$. Determining b is a bit trickier. Here we have to use that $y_i \left(w^\top x_i + b \right) - 1 \geq 0$ for all i so we have

$$b \begin{cases} \geq 1 - w^\top x_i & \text{if } y_i = 1, \\ \leq -1 - w^\top x_i & \text{if } y_i = -1, \end{cases}$$

Furthermore the inequalities are equalities at the support vectors. Hence

$$b = \begin{cases} \min\limits_{i,y_i=1} \left\{ 1 - w^\top x_i \right\} & = \min\limits_{i,y_i=1} \left\{ 1 - \sum_{j=1}^m \lambda_j y_j x_j^\top x_i \right\} \\ \max\limits_{i,y_i=-1} \left\{ -1 - w^\top x_i \right\} & = \max\limits_{i,y_i=-1} \left\{ -1 - \sum_{j=1}^m \lambda_j y_j x_j^\top x_i \right\}. \end{cases}$$

So finally to classify z we calculate $b + \frac{1}{2}\sum_{i=1}^m \lambda_i y_i x_i^\top z$ (which in this case is $w^\top z + b$) and return whether it is positive or negative.

Why should we prefer the new dual formulation over the one we already had? First of all the primal formulation has $n+1$ variables, where n denotes the dimension in which we want to separate. The dual program on the other hand has m variables, one for each test vector. In applications where the dimension of the vector space exceeds the number of test points significantly, the dual formulation can be solved faster than the primal one. This naturally occurs in image processing where one can for example describe a picture as vector with multiple dimensions (RGB) per pixel.

The second main advantage is that both the dual problem and the classification only need information about the scalar products $x_i^\top x_j$ of the test vectors as well as the scalar products $z^\top x_i$ of the vector to be classified with the test vector. This is very useful when applying SVM to non-linear classification problems as done in section 2.4. Recall that there we were given non-linearly classifiable points and the trick was to first map them to a higher dimensional vector space and separate them there. This was called a *feature map*. So we used a mapping

$$\Phi\colon \mathbb{R}^n \to \mathbb{R}^N$$

and classified $\Phi(z)$ with respect to $\Phi(x_i)$. Now in the dual formulation of the transformed problem we will only consider scalar products like $\Phi(x_i)^\top \Phi(x_j)$. Hence if there is a closed formula for scalar products of Φ values we can skip going to a higher dimension. We would instead use the *kernel*

$$k\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad (x,y) \mapsto \Phi(x)\cdot\Phi(y).$$

Consider for example the norm feature map

$$\Phi\colon \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}, \tag{3.13}$$

then its kernel would just be $k(x,y) = (x^\top y)^2$. So instead of solving a three dimensional problem we can stay in the two dimensional setup. In general each kernel is induced by a feature map but they may be hard to give explicitly or high dimensional. For example a common kernel, the *Gaußian kernel* is given by

$$k(x,y) = \mathrm{e}^{-\frac{\|x-y\|^2}{2\sigma^2}}$$

for any $\sigma \in \mathbb{R}_{>0}$. The feature map inducing this kernel would have to map into an infinite dimensional vector space. So the dual program allows for more flexibility when extending SVM to non-linear settings.

3.9 Exercises

Exercise 3.1 Derive an explicit description of the dual problem for the linear program in canonical form.

Exercise 3.2 — Programming Exercise.
1. Use the convex program we constructed in section 3.8.1 to implement the SVM via the dual point of view. Your program should in particular be able to classify a new point x. Test it on the data files given on the webpage.
2. Extend your program such that it can process kernels. You can find further test data for kernels on the webpage as well.

Hint: You can directly implement part 2 and use that part 1 is the special case for a particular kernel.

Exercise 3.3 Consider the following game played by two players A and B. There is a fixed payout matrix $P \in \mathbb{R}^{m\times n}$, A chooses a row $k \in \{1,\dots,m\}$ and simultaneously B chooses a column $l \in \{1,\dots,n\}$. Then A pays the ammount P_{kl} to B (if P_{kl} is negative interpret this as B pays $|P_{kl}|$ to A.)

Both players incorporate a randomized strategy, meaning that A chooses a random distribution on $\{1,\dots,m\}$, hence $a_1,\dots,a_m \geq 0$ with $\sum_{i=1}^m a_i = 1$. They then choose the row at random with these probabilities. Similarly B uses a random distribution on $\{1,\dots,n\}$ to determine which column they choose. The goal of A is thus to minimize the expectation of the payment P_{kl} while B tries to maximize it.

Now consider two scenarios. First assume that A knows the distribution B will use and they can adapt their distribution accordingly. Formulate the problem of B finding a strategy such that no matter how A adapts the expected payment is minimized as a linear program (B). So in this scenario A has an advantage and B wants to find a solution minimizing A's exploit of that.

In the second scenario the roles are reversed and B knows the strategy of A. Formulate finding the strategy A should use as a linear program (A).

For both linear programs formulate the dual. What can you observe? And what does this mean for the game and both scenarios?

Exercise 3.4 — Max-Min Inequality. Let X, Y be sets and $f : X \times Y \to \mathbb{R}$ *any* function. Proof that

$$\sup_{x\in X}\inf_{y\in Y} f(x,y) \leq \inf_{y\in Y}\sup_{x\in X} f(x,y)$$

always holds.

R The above exercise has a game theoretic interpretation that makes it easier to remember. Suppose there is a game where you want to maximize a certain value and your opponent wants to minimize it. Then the above inequality states that you will always perform better if you know your opponents move i.e. he has to move first.

Exercise 3.5 — Conjugate Function. In this exercise we will look more closely at the dual function defined in Example 3.1.8. For a function $h\colon D \to \overline{\mathbb{R}}$, where $D \subseteq R^n$ and $\overline{\mathbb{R}} := \mathbb{R}\cup\{-\infty,\infty\}$, we define its *conjugate function* as

$$h^*\colon D \to \overline{\mathbb{R}},$$
$$h^*(\Phi) := \sup_{x\in D} \Phi^\top x - h(x).$$

We use the common convention $0\cdot\pm\infty = 0$.

1. Proof the following basic properties
   (a) The conjugacy is *order-reversing*, i.e. for $f, g\colon D \to \overline{\mathbb{R}}$ it holds that

   $$f \leq g \Rightarrow f^* \geq g^*.$$

   (b) If $h\colon D \to (-\infty, \infty]$ and h not constantly ∞ we have

   $$h(x) + h^*(\Phi) \geq \Phi^\top x$$

   for every $\Phi, x \in D$ (be careful with edge cases since $\infty - \infty$ is not defined).
2. Let $f\colon \mathbb{R}^n \to (-\infty, \infty]$ and $g\colon \mathbb{R}^m \to (-\infty, \infty]$, and let $A\colon \mathbb{R}^n \to \mathbb{R}^m$ be a linear map. We define

   $$\overline{\mathbb{R}} \ni p := \inf_{x \in \mathbb{R}^n} f(x) + g(Ax) \text{ and}$$
   $$\overline{\mathbb{R}} \ni q := \sup_{\Phi \in \mathbb{R}^m} -f^*(A^*\Phi) - g^*(-\Phi).$$

   Here A^* denotes the adjoint operator of A i.e. it fulfils $\langle Ax, y\rangle = \langle x, A^*y\rangle$ for all $x \in R^n, y \in R^m$. Prove that $p \geq q$.
3. Conclude that for $b \in \mathbb{R}^m, f\colon \mathbb{R}^n \to (-\infty, \infty]$ and a linear map $A\colon \mathbb{R}^n \to \mathbb{R}^m$ the weak duality

   $$\inf_{x \in \mathbb{R}^n} \{ f(x) \mid Ax = b \} \geq \sup_{\Phi \in \mathbb{R}^m} b^\top \Phi - f^*(A^*\Phi)$$

   holds.
4. Now suppose that f and g are convex and satisfy

   $$A(\operatorname{dom} f) \cap \operatorname{cont} g \neq \emptyset,$$

   where $\operatorname{cont} g := \{ x \in \mathbb{R}^m \mid g(x) \in \mathbb{R} \text{ and } g \text{ is continuous at } x \}$. Prove that in this case

   $$p = q$$

   holds.
   Hint: You may use without proof that under the conditions above, the function $h\colon \mathbb{R}^m \to \overline{\mathbb{R}},\ h(u) := \inf_{x \in \mathbb{R}^n} f(x) + g(Ax + u)$ has a subgradient Φ in 0, i.e. there exists $\Phi \in \mathbb{R}^m$ such that $h(u) \geq h(0) + \Phi^\top u$ at every $u \in \mathbb{R}^m$.

Exercise 3.6 — Slater's Condition and Compactness. Prove that Slater's condition (Theorem 3.5.1) holds if and only if there exists an $x^* \in \mathbb{P}$ for which the level set

$$\left\{ \lambda \in \mathbb{R}^m_{\geq 0} \mid -L(x^*, \lambda, \nu) \leq \alpha \right\}$$

is compact for every $\alpha \in \mathbb{R}$ and every $\nu \in \mathbb{R}^\ell$ and $h(x^*) = 0$ holds.

Exercise 3.7 — Explicit Duals. In this exercise we will derive explicit duals as done in Example 3.1.7. Therefore we look at mathematical problems of the form

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &\text{s.t. } Ax \leq b, \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Derive the explicit dual problem for the given $f\colon \mathbb{R}^n \to \mathbb{R}$.

i) $f(x) = \frac{1}{2}x^\top Cx$ for a given symmetric positive definite (see Definition 9.1.3) $C \in \mathbb{R}^{n\times n}$ (Quadratic Program),

ii) $f(x) = \sum_{j=1}^n p(x_j)$ for a given function $p\colon \mathbb{R} \to \mathbb{R}$ (separable problem),

iii) $f(x) = c^\top x + \varepsilon \operatorname{lb}(x)$ where $c \in \mathbb{R}^n$ and $\varepsilon > 0$. Here the function $\operatorname{lb}\colon \mathbb{R}^n \to (-\infty, \infty]$ is the so called log barrier,

$$\operatorname{lb}(x) := \begin{cases} -\sum_{j=1}^n \log(x_j) & \text{, if } x > 0 \\ +\infty & \text{, otherwise.} \end{cases}$$

R The log barrier simulates the constraint $x \geq 0$ but is also *tries to push the solution away from the border*.

Exercise 3.8 — Extension of Slater's condition. Prove the following extension of Slater's condition:

Theorem For any convex program if there exists a point $x^* \in \operatorname{int}(\mathscr{P})$ with $g_i(x^*) < 0$ for all non-affine g_i, $g_i(x^*) \leq 0$ for all affine g_i and $h_j(x^*) = 0$, then strong duality holds and the dual optimum value is attained.

3.10 Solutions

Solution of Exercise 3.3

Let $a = (a_1, \dots, a_m) \in \mathbb{R}^m_{\geq 0}$ with $\mathbf{1}^\top a = 1$ denote the distribution used by A and $b = (b_1, \dots, b_n)$ the one used by B. Then the expected payment is

$$p = a^\top P b.$$

The goal of A is to minimize p while B aims to maximize it. Consider the first scenario in which A knows b and adapts a. As a has to be normalized and non-negative we know:

$$p = a^\top (Pb) = \sum_{i=1}^m a_i (Pb)_i \geq \sum_{i=1}^m a_i \min_j (Pb)_j = \min_j (Pb)_j$$

and this is an equality if A chooses a_j to be one for the index j where the maximum is attained. So the optimization problem B has to can be written in the following two forms:

$$\begin{array}{lll} \max & \min\ (Pb)_j & \\ s.t. & b \geq 0, & \\ & \mathbf{1}^\top b = 1, & \end{array} \qquad \begin{array}{lll} \max & t & \\ s.t. & t \leq P_j b & \text{for all } 1 \leq j \leq m, \\ & b \geq 0, & \\ & \mathbf{1}^\top b = 1. & \end{array} \qquad \text{(B)}$$

Similarly the problem for A in the second scenario is to minimize $p = a^\top P b = b^\top P^\top a$ for the optimal choice of b. As above B can choose b such that $p = \max (P^\top a)_j$, so the problem for A is The dual of both (A) and (B) can be computed straight forwardly: For (B) let β_j be the dual variable

$$\begin{array}{lll} \min & \max\ (P^\top a)_j & \\ s.t. & a \geq 0, & \\ & \mathbf{1}^\top a = 1, & \end{array} \qquad \begin{array}{lll} \min & t & \\ s.t. & t \geq P_j^\top a & \text{for all } 1 \leq j \leq n, \\ & a \geq 0, & \\ & \mathbf{1}^\top a = 1. & \end{array} \qquad \text{(A)}$$

of the j-th inequality and let ν be the dual variable of the equality:

$$\begin{array}{lll} \min & \nu & \\ s.t. & \beta \geq 0, & \\ & \mathbf{1}^\top \beta = 1, & \\ & \nu \geq P^\top \beta. & \end{array} \qquad \text{(B*)}$$

Similarly with α_j and μ for (A):

$$\begin{array}{lll} \max & \mu & \\ s.t. & \alpha \geq 0, & \\ & \mathbf{1}^\top \alpha = 1, & \\ & \mu \leq P\alpha. & \end{array} \qquad \text{(A*)}$$

Now we can see that (A) and (B*) as well as (B) and (A*) are equivalent. By strong duality for linear programs we deduce that the optimal values of (A) and (B) agree. In particular knowing the strategy of the oponent does not give either player an advantage!

Solution of Exercise 3.4

We denote $h(x) \coloneqq \inf_{y\in Y} f(x,y) \in \mathbb{R} \cup \{-\infty\}$ for $x \in X$. Due to $f(x,y) \geq h(x)$ for all $x \in X, y \in Y$ we get for all $y \in Y$ that

$$\sup_{x\in X} h(x) \leq \sup_{x\in X} f(x,y).$$

Therefore also

$$\sup_{x\in X} h(x) \leq \inf_{y\in Y} \sup_{x\in X} f(x,y),$$

which proves the statement by the definition of h.

Solution of Exercise 3.5

We first note, that for nonempty D the conjugate h^* never takes the value $-\infty$ if h is finite somewhere. We now start with proving the properties. Note that both properties have all quantors over D and hence are trivially true for $D = \emptyset$. Therefore let $D \neq \emptyset$ for exercise 1.

1) (a) Let $f, g \colon D \to \overline{\mathbb{R}}$ with $\forall x \in D \colon f(x) \leq g(x)$. Then for $\Phi \in D$

$$\begin{aligned} f^*(\Phi) &= \sup_{x\in D} \Phi^\top x - f(x) \\ &\geq \sup_{x\in D} \Phi^\top x - g(x) \\ &= g^*(\Phi) \end{aligned}$$

where the inequality holds since we lower bound each element of the set.

1) (b) Since we assume that D is nonempty, neither h nor h^* take the value $-\infty$ and we can calculate as usual. Furthermore, if either $h(x)$ or $h^*(\Phi)$ is ∞, the statement trivially holds. In the finite case the statement does however follow immediately since we can subtract $h(x)$ on both sides and use the definition of sup.

2) We first deal with the case where f or g is constantly ∞. In this case we get $p = \infty$ and hence the inequality always holds. For the non-degenerated case first note that, since $\mathbb{R}^n$ and $\mathbb{R}^m$ are nonempty, we get $p < \infty$ and $q \geq -\infty$. Using $1)(b)$ for $x \in \mathbb{R}^n$ and $A^*\Phi \in \mathbb{R}^m$, it further holds that

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

On the other hand $1)(b)$ applied to g gives us

$$g(Ax) \geq -g^*(-\Phi) - \Phi^\top Ax.$$

Summing up, we have for any $x \in \mathbb{R}^n$ and any $\Phi \in \mathbb{R}^m$ that

$$\begin{aligned} f(x) + g(Ax) &\geq -f^*(A^*\Phi) + (A^*\Phi)^\top x - g^*(-\Phi) - \Phi^\top Ax \\ &= -f^*(A^*\Phi) - g^*(-\Phi) + (A^*\Phi)^\top x - \Phi^\top Ax \\ &= -f^*(A^*\Phi) - g^*(-\Phi) + \Phi^\top (Ax) - \Phi^\top Ax \\ &= -f^*(A^*\Phi) - g^*(-\Phi), \end{aligned}$$

which proves the statement.

3) We define the indicator function $g\colon \mathbb{R}^m \to \{0,\infty\}$,

$$g(x) := \begin{cases} 0 & \text{, if } x = b \\ \infty & \text{, otherwise.} \end{cases}$$

Then we get $g^*(\Phi) = \Phi^\top b$ and the claim follows by applying 2).

4) We first note that $h(0) = p$ and calculate for any $u \in \mathbb{R}^m, x \in \mathbb{R}^n$ using the hint applied to $-\Phi$:

$$\begin{aligned} h(0) &\leq h(u) + \Phi^\top u \\ &\leq f(x) + g(Ax+u) + \Phi^\top u \\ &= \left(f(x) - (A^*\Phi)^\top x\right) + \left(g(Ax+u) - (-\Phi)^\top (Ax+u)\right). \end{aligned}$$

If we now first take the infimum in regards to u and then x we get

$$p = h(0) \leq -f^*(A^*\Phi) - g^*(-\Phi) \leq q \leq p$$

hence it must be equality everywhere and the claim follows.

Solution of Exercise 3.6

First suppose that $x^* \in \mathbb{P}$ fulfils slater's condition. Then by definition $h(x^*) = 0$ and $A := \left\{ \lambda \in \mathbb{R}^m_{\geq 0} \mid -L(x^*, \lambda, \nu) \leq \alpha \right\}$ is closed. Therefore we need to show that A is bounded to obtain the claim. Suppose there exist an unbounded sequence $(\lambda^{(n)})_{n\in\mathbb{N}}$ in A. Due to the equivalence of norms we get $\left\|\lambda^{(n)}\right\|_1 \to \infty$ and if we define $u := \max_{i\in[m]} g_i(x^*) < 0$ we get for any $\nu \in \mathbb{R}^\ell$ that

$$\begin{aligned} L(x^*, \lambda^{(n)}, \nu) &= f(x^*) + \sum_{i=1}^m \lambda_i^{(n)} g_i(x^*) + \sum_{j=1}^\ell \nu_j h_j(x^*) \\ &\leq f(x^*) + \sum_{i=1}^m \lambda_i^{(n)} u \\ &= f(x^*) + u \left\|\lambda^{(n)}\right\|_1. \end{aligned}$$

Since $u < 0$ we get that $L(x^*, \lambda^{(n)}, \nu) < \delta$ for any $\delta := -\alpha \in \mathbb{R}$ and large enough n. This contradicts $\lambda^{(n)} \in A$ proving the claim with the same x^*.

Now suppose the other direction holds, i.e. there exists $x^* \in P$ for which the set A is compact for any α, ν and $h(x^*) = 0$. Again we conduct a proof by contradiction by assuming $g_i(x^*) \geq 0$ for some $i \in [m]$ (i.e. assuming the same x^* is not a slater point). We set $\lambda^{(n)} := ne^{(i)}$ where $e^{(i)} \in \mathbb{R}^m$ is the i-th unit vector (i.e. $e^{(i)}_j = \delta_{i,j}$) and $\alpha := -L(x^*, 0, 0) = -f(x^*)$. This gives

$$\begin{aligned} \alpha &= -f(x^*) \\ &\geq -f(x^*) - n g_i(x^*) \\ &= -L(x^*, \lambda^{(n)}, 0). \end{aligned}$$

This shows that $\lambda^{(n)} \in A$ for every $n \in \mathbb{N}$, for our choice of α and ν. Since $\left\|\lambda^{(n)}\right\|_2 = n$ we get that A is not compact, yielding the contradiction we were looking for. This concludes the proof.

Solution for Exercise 3.7

In order to express the dual problem we have to start with the Lagrangian which, for our problem, can be written as

$$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \left(a_i^\top x - b_i \right)$$

where $a_i^\top$ is the i-th row of A. This gives us the Lagrange dual function

$$\widehat{L}(\lambda) = \inf_{x \in \mathbb{R}^n} f(x) + \sum_{i=1}^{m} \lambda_i \left(a_i^\top x - b_i \right)$$

and the lagrange dual program

$$\begin{aligned} &\sup \widehat{L}(\lambda), \\ &\quad \lambda \in \mathbb{R}^m_{\geq 0}. \end{aligned}$$

We now start calculating the explicit expressions.

i) Let $C \in \mathbb{R}^{n \times n}$ symmetric positive definite. Then we get

$$L(x,\lambda) = \frac{1}{2} x^\top C x + \lambda^\top (Ax - b)$$

which is differentiable and hence we calculate

$$\partial_x L(x,\lambda) = Cx + A^\top \lambda.$$

This gives us the stationary point $x = -C^{-1}A^\top \lambda$. Since C is positive definite, this solution is a global minimum and we get Lagrange dual function

$$\begin{aligned} \widehat{L}(\lambda) &= \frac{1}{2} \lambda^\top A C^{-1} A^\top \lambda + \lambda^\top \left(-AC^{-1}A^\top \lambda - b \right) \\ &= -\frac{1}{2} \lambda^\top A C^{-1} A^\top \lambda - \lambda^\top b. \end{aligned}$$

Finally the dual problem is given by

$$\sup_{\lambda \geq 0} -\frac{1}{2} \lambda^\top A C^{-1} A^\top \lambda - \lambda^\top b$$

iii) Let $\varepsilon > 0$. As before we get

$$L(x,\lambda) = c^\top x + \varepsilon \operatorname{lb}(x) + \lambda^\top (Ax - b)$$

and

$$\begin{aligned} \widehat{L}(\lambda) &= \inf_x L(x,\lambda) \\ &= \begin{cases} \inf_x g(x) \coloneqq c^\top x - \varepsilon \sum_{j=1}^n \log(x_j) + \lambda^\top (Ax - b), & \text{if } x > 0 \\ +\infty, & \text{otherwise.} \end{cases} \end{aligned}$$

Hence let $x > 0$ in which case we may differentiate and get

$$\begin{aligned} g'(x) &= -\varepsilon \sum_{j=1}^{n} \frac{1}{x_j} e_j + (c + A\lambda) \\ g''(x) &= \varepsilon \begin{pmatrix} \frac{1}{x_1^2} & & \\ & \ddots & \\ & & \frac{1}{x_n^2} \end{pmatrix} \succ 0. \end{aligned}$$

Now let $a_j^\top$ denote the j-th row of A. Then the minima is attained when $x > 0$ and

$$\begin{aligned}\frac{\varepsilon}{x_j} &= (c + A\lambda)_j \\ &= c_j + a_j^\top \lambda,\end{aligned}$$

for every $j \in [n]$. Rewriting gives

$$x_j = \frac{\varepsilon}{c_j + a_j^\top \lambda}.$$

For the case $\left(c_j + a_j^\top \lambda\right) \leq 0$, we can push the value to $-\infty$ by choosing $x_j \to \infty$. Therefore the lagrange dual program is given by

$$\begin{aligned}\max\ & \varepsilon - \varepsilon \sum_{j=1}^{n} \log\left(\frac{\varepsilon}{c_j + a_j^\top \lambda}\right) - \lambda^\top b \\ \text{s.t. } & A^\top \lambda + c > 0 \\ & \lambda \in \mathbb{R}^n_{\geq 0}.\end{aligned}$$

Part Two

4. Unconstrained optimization

In this chapter, we study methods to solve an unconstrained convex optimization problem under the assumption that the objective function is twice continuously differentiable. We discuss two classes of methods to solve such problems. The first one consists in finding a local (global) optimal solution analytically, presented in Section 4.1. Though very powerful, this method is not always applicable, especially when optimizing a function for which not all the information is provided. The second method is based on an algorithmic approach. Since we work on twice differentiable functions, we can exploit the fact that for every given point, we can easily detect if there are points in the neighborhood that improve the objective value. Given an initial point, the algorithm iteratively improves the solution until it either reaches an optimal solution or is sufficiently close to it. The class of such methods are called descent methods and are discussed in Section 4.2. The method, at its heart, is fairly simple. In every iteration, it chooses a direction in which an incremental change should happen and then decides on the size of the step in this direction. In this chapter, we will investigate both procedures carefully. First, in Section 4.3 we present two most popular methods to choose a step size, a more efficient method — exact line search, that not always can be applied since determining an exact step size might not always be possible, and a more tractable method — backtracking line search. Next, we present several methods to determine the direction of the step. The two most popular methods are Gradient Descent and Newton's method, discussed in Sections 4.5 and Section 4.6, respectively. We show that both methods have a very natural motivation that originates from similar arguments. The latter is an improved version of the former. In those two sections, we focus on the convergence analysis. We show that under some assumptions, both methods can efficiently solve unconstrained convex optimization problems. However, the convergence depends on some parameters that first might not always be known; second, for some choice of parameters, the convergence of the descent method is much slower. For this reason, in Section 4.7 we study self-concordant functions, a class of functions for which the convergence of Newton's method is independent of the parameters mentioned above.

4.1 Analytic method

In this section, we discuss the first type of method that allows solving an unconstrained convex optimization problem under the assumption that the function is twice continuously differentiable. As mentioned in the introduction, this method is applicable only in some specific situations when we have full information about the objective function and analytic methods to compute local (global) minimum exist. More formally, in this section, we study the following optimization problem

$$f^* = \min f(x),$$

where $f: \mathbb{R}^n \to \mathbb{R}$ is convex and twice continuously differentiable. By f^* we denote the value of the optimal solution. We assume that there is only one optimal solution. Since the function is differentiable and convex, a necessary and sufficient condition for a point x^* to be optimal is

$$\nabla f(x^*) = 0.$$

Finding such a point boils down to solving a system of n equations in n variables. In some special, rare cases, the system can be solved analytically. One such case is presented in the example below.

■ **Example 4.1.1** Consider the following general quadratic minimization problem of the form

$$\min \frac{1}{2}x^\top Px + q^\top + r,$$

for $P \in \mathbb{R}^{n\times n}$ being a positive semidefinite matrix, $q \in \mathbb{R}^n$ and $r \in \mathbb{R}$. The optimality condition reduces to the form $Px^* + q = 0$ which is a set of n linear equations in n variables. The system has a unique solution if the matrix P is positive definite (i.e., P^{-1} exists) that is $x^* = -P^{-1}q$. When P is not positive definite (but still positive semidefinite, i.e., has the smallest eigenvalue equal to 0) any solution of the system $Px^* + q = 0$ is optimal, and if there are no solutions to that system, the problem is unbounded. ■

4.2 Descent method

In most cases, unfortunately, it is not possible to compute an analytic, optimal solution of the system given by the optimality conditions. In that case, we follow an algorithm method that iteratively improves a solution. The algorithm will be very simple. Given an initial point, it will iteratively improve the solution by first deciding on the descent direction and then on the step size to be made in this direction. More precisely, for a problem of minimizing a function $f: U \subseteq \mathbb{R} \to \mathbb{R}$ the algorithm will produce a sequence of points

$$x^{(0)}, x^{(1)}, \ldots \in U$$

such that

- $f(x^{(k+1)}) < f(x^{(k)})$ for every $k \in \mathbb{N}$,
- and $f(x^{(k)}) \to f^*$ as $k \to \infty$.

Since, as mentioned above, the algorithm will produce a sequence of points iteratively based on the descent direction and the step size, for $k \in \mathbb{N}$, we write

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)},$$

where $t^{(k)}$ is the step size and $\Delta x^{(k)}$ is the descent direction in the k'th step.

In practice, we terminate the algorithm when the solution is close enough to the optimal one. Otherwise, the algorithm might never terminate. Moreover, this stopping criterion arises naturally from the encoding precision in computers and, in that sense, is not very restrictive. We are ready to present the algorithm.

In the remainder of this section, we will analyze the descent method for various descent steps (Algorithm 1: line 4) and descent directions (Algorithm 1: line 3). We start by presenting two methods to determine a decent step in the Descent method.

Algorithm 1: Descent method

Input: starting point $x^{(0)} \in U$
Output: point close to optimal

$k \leftarrow 0$
repeat
  choose a descent direction $\Delta x^{(k)}$
  choose a descent step $t^{(k)}$
  $x^{(k+1)} \leftarrow x^{(k)} + t^{(k)} \Delta x^{(k)}$
  $k \leftarrow k+1$
until *stopping criteria is met*;
return $x^{(k)}$

4.3 Descent method — step size

In this section we discuss two methods to choose a step size (Step 2 in the descent method).

Exact line search algorithm

In the exact line search algorithm the step size is chosen to be the maximum possible step in the descent direction. More precisely, it takes the following step

Algorithm 2: Exact line search

Input: a descent direction Δx at point x
Output: a descent step t

$t = \arg\min_{t \geq 0} (x + t\Delta x)$
return t

The exact line search algorithm outputs the maximum possible step size in the descent direction. That is, it provides the maximum reduction in the value of function f in this direction. However, the exact line search algorithm is an optimization problem in one variable (t), and running the algorithm might be time-consuming. The algorithm can be used if solving the optimization problem is computationally cheap compared to choosing a descent direction. In other cases, the gain from the optimal descent step might not compensate for the computational effort. Last but not least, since exact line search is a convex optimization problem, we might be able to find an analytic solution for this problem, as discussed in Section 4.1.

■ **Example 4.3.1** An example of exact line search for given $f, \Delta x$ can be seen in Figure 4.1. Note that $f(x + t^* \Delta x)$ is indeed a minimum and Δx a descent direction. ■

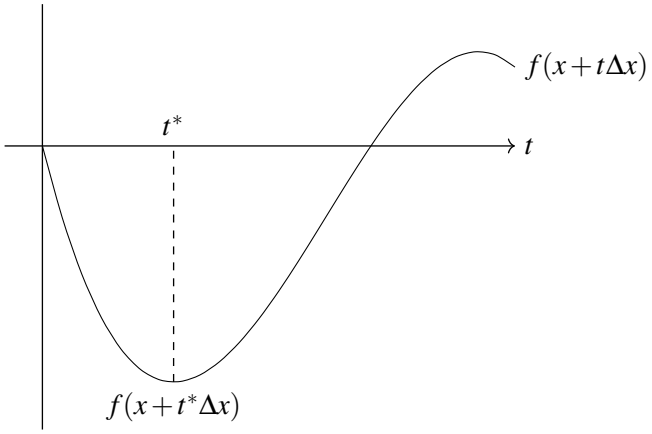
Backtracking line search algorithm

Since, in many cases, it is not possible or not computationally reasonable to perform an exact line search algorithm, some methods perform an inexact search. The idea behind such methods is to iteratively search for a descent step such that, on the one hand, we achieve a significant decrease in the value of the function f; on the other hand, we want to be sure not to go too far, since this can have the opposite effect.

Note that for any $x \in U$, for convex differentiable function (which implies the function is continuously differentiable) and $t \geq 0$ we have the following lower bound on the value of the function f (see Theorem 9.2.5)

$$f(x + t\Delta x) \geq f(x) + t \nabla f(x)^\top \Delta x$$

Figure 4.1: Visualization of the exact line search algorithm.



However, it is difficult to estimate a reasonable size of the step t based on this lower bound. A remedy for this situation is to estimate the size of the step using the following upper bound on the function f. Let $\alpha \in (0,1)$, then for $t \in [0,\widehat{t}]$, where $\widehat{t}$ is a small enough constant, the depends on the choice of alpha, we have

$$f(x+t\Delta x) \leq f(x) + \alpha t \nabla f(x)^\top \Delta x.$$

Think of the above inequality so that we sacrifice the descent direction (make it less steep), but for some small t, we obtain an upper bound on $f(x+t\Delta x)$. The idea of the backtracking line search algorithm is to balance out the value of α and the position of a point that is desirably close enough to $\widehat{t}$ such that we make measurable progress in the descent algorithm. The backtracking algorithm is presented in Algorithm 3.

Algorithm 3: Backtracking line search

Input: a descent direction Δx at point x, constants $\alpha \in (0, 0.5)$ and $\beta \in (0,1)$
Output: a descent step $t \geq \min\{1, \beta\widehat{t}\}$

```
t ← 1
while f(x + tΔx) > f(x) + α∇f(x)^⊤ (t · Δx) do
    t ← βt
end
return t
```

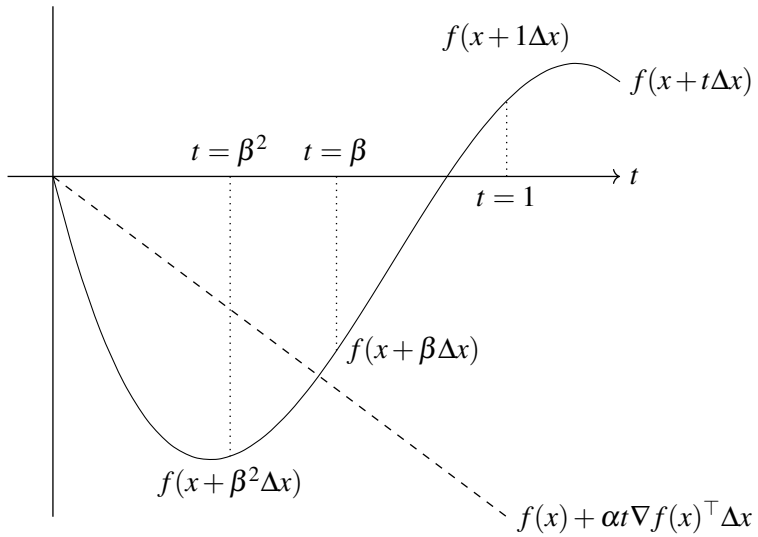
Note that the stopping condition is satisfied for every $t \in [0,\widehat{t}]$. Since, we start with $t = 1$ and in every step we scale down t by a factor of β the backtracking line search outputs $t \geq \min\{1, \beta\widehat{t}\}$. Example 4.3.2 visualizes the steps of the backtracking line search algorithm.

■ **Example 4.3.2** Figure 4.2 shows an example application of the backtracking algorithm. The dashed line shows the bound under which we must end up and the intersection with $f(x+t\Delta x)$ marks $\widehat{t}$. In this example $t = \beta^2$ is returned by Algorithm 3. ■

4.4 Descent method — step direction

In this section, we discuss methods to choose a descent step. There are many methods available. We focus on the two most popular methods that share the same motivation. The methods we discuss

Figure 4.2: Example of the Backtracking line search Algorithm. In this example we move from the right ($t = 1$) to the left until we are below the threshold $f(x) + \alpha t \nabla f(x)^\top \Delta x$.



in the following two subsections are the Gradient descent method and Newton's method. We start with the following motivation.

In Section 4.1 we mentioned that the best method, though not always applicable, is to compute the minimum of the convex function analytically. We do not have to perform an iterative search and output an approximate optimal point in such a case. Instead, we directly get an optimal solution. Clearly, it is not always possible to apply the analytic method. However, what is possible is to approximate the function for a given point x with a simpler function and then compute the minimum of the approximate function. This can be used in the iterative algorithm to find a good descent direction at each iterative step of the algorithm.

To do this end we use the Taylor expansion (for some more details see Theorem 9.2.6, 9.2.7 and 9.2.8 in the Appendix — Chapter 9) of a twice continuously differentiable function at point x to obtain the first and the second Taylor expansion of the form:

$$f(x+v) \approx f(x) + \nabla f(x)^\top v$$

and

$$f(x+v) \approx f(x) + \nabla f(x)^\top v + \frac{1}{2} v^\top \nabla^2 f(x) v$$

respectively. By the necessary optimality conditions (Preposition 9.2.9) we get a descent direction for the gradient descent method of the form

$$\Delta x_{GD} = -\nabla f(x)$$

and for Newton's method of the form

$$\Delta x_N = -\frac{\nabla f(x)}{\nabla^2 f(x)}.$$

We make several important observations. First, note that for both descent directions we get $\nabla f(x)^\top \Delta x_{GD} < 0$ and $\nabla f(x)^\top \Delta x_N < 0$, unless x is an optimal solution, where the first inequality holds trivially and the second inequality holds since $\nabla^2 f(x)$ is a positive definite matrix, called a

Hessian (see Chapter 9). Thus we make progress in the descent method, unless x is an optimal solution.

The second observation is that if the domain U of the function f is not $\mathbb{R}^n$ we must be careful when performing the backtracking steps to be sure we operate within the domain of f. In practice usually we keep multiplying t by β until $x+t\Delta x$ is inside U.

Finally, we want to point out that Newton's method, based on the second-order Taylor expansion, gives a close to an optimal approximation of the minimizer of the function f, for f having a close to quadratic behavior. In particular, it gives the minimizer of f when f is a convex quadratic function. In this case, Newton's method and the analytic method coincide.

■ **Example 4.4.1** We again look at the function from the previous examples which is given by

$$f(x) = -\frac{5}{12}x^3 + \frac{15}{8}x^2 - 2x.$$

In this case the gradient is given by

$$\nabla f(x) = -\frac{5}{4}x^2 + \frac{15}{4}x - 2,$$

and the Hessian is

$$\nabla^2 f(x) = -\frac{5}{2}x + \frac{15}{4}.$$

Therefore the first order approximation in $x = 0$ is given by

$$f(x+v) \approx T_1(v) := -2v,$$

and the second order by

$$T_2(v) := -2v + \frac{15}{4}v^2.$$

The descent directions are $\Delta x_{\text{GD}} = 2$ and $\Delta x_{\text{N}} = \frac{8}{15}$. This situation can be seen in Figure 4.3. One can imagine that if f was closer to a second order polynomial, the newton step would be even closer to the local minimum.

■

4.5 Gradient descent method

In this section we study the Gradient descent method. Here we assume that the function f we want to minimize is strongly convex.

Definition 4.5.1 — strongly convex/concave function. Let $X \subseteq \mathbb{R}^n$ be a convex set. A differentiable function $f : X \to \mathbb{R}$ is *strongly convex* with some constant m if for all $x, y \in X$ and $\lambda \in [0,1]$ it holds that

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) - \frac{\lambda(1-\lambda)m}{2}\|x-y\|^2.$$

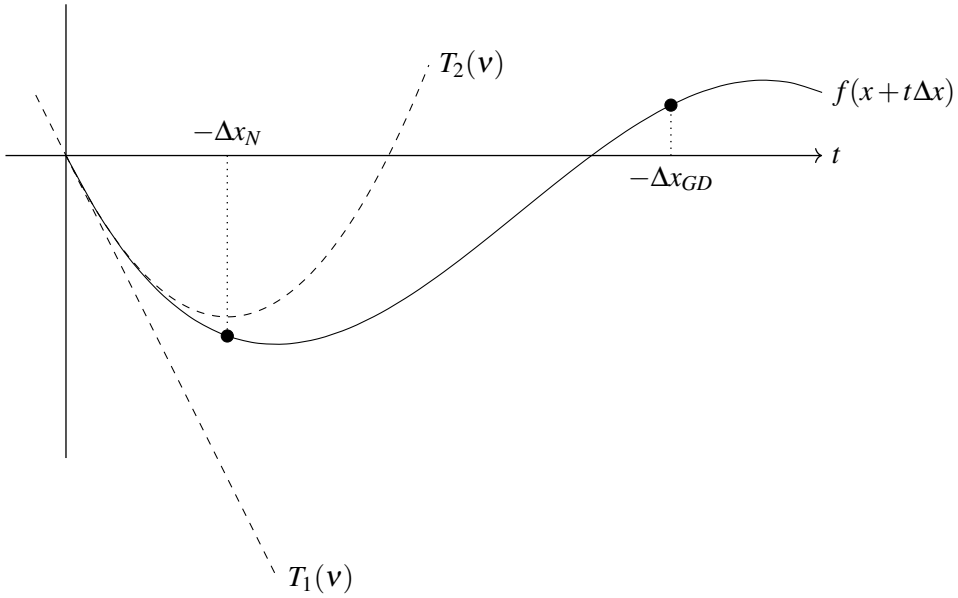
Moreover, if a function f is such that $-f$ is strongly convex with constant m we call it *strongly concave* function with constant m.

Strongly convex functions have many other equivalently definitions, some of them are summarized in the following proposition (some of them require the function to be twice differentiable).

Proposition 4.5.2 Let f be a twice differentiable function. The following statements are equivalent:

1. function $f : U \subseteq \mathbb{R}^n \to \mathbb{R}$ is strongly convex with constant m ,

Figure 4.3: Visualization of the Gradient and Newton method. Here T_1, T_2 denote the taylor approximations and $\Delta x_N, \Delta x_{GD}$ the corresponding descent directions.



2. $f(x) - m/2\,\|x\|^2$ is convex ,
3. $\nabla^2 f(x) \succeq mI$, for all $x \in U$,
4. $f(y) \geq f(x) + \nabla f(x)^\top (y-x) + m/2\,\|y-x\|^2$ for all $x, y \in U$.

Proof.

(1) $\Leftrightarrow$ (2). Equivalence follows directly from the definition of convexity, Definition 1.1.7.

(2) $\Leftrightarrow$ (3). To prove $\Rightarrow$ note that since $f(x) - m/2\,\|x\|^2$ is convex and twice differentiable thus its Hessian is positive semidefinite, with the $i, j \in [n]$ entry equal

$$\frac{\partial}{\partial x_i}\frac{\partial}{\partial x_j}\left(f(x) - \frac{m}{2}\|x\|^2\right) = \frac{\partial}{\partial x_i}\frac{\partial}{\partial x_j} f(x) - mI,$$

which implies that the Hessian of f satisfies $\nabla^2 f(x) - mI \succeq 0$ which implies the claim. To prove the $\Leftarrow$ we just reverse the steps.

(2) $\Leftrightarrow$ (4). Equivalence follows directly from the first-order conditions for convexity of differentiable functions (Theorem 9.2.5), i.e., $g(x)$ is convex iff $g(y) \geq g(x) + \nabla g(x)^\top (y-x)$ for all $x, y \in U$. ■

We can use the (4) from Proposition 4.5.2 to bound the distance $f(x) - f^*$ and to derive a stopping criterion for a Gradient descent algorithm. To bound the distance $f(x) - f^*$ note that the right hand side of (4), for a fixed x, is a convex quadratic function of y, setting the gradient with respect to y to zero we get that $\tilde{y} = x - 1/m\nabla f(x)$ minimizes the rand hand side, thus we get

$$\begin{aligned} f(y) &\geq f(x) + \nabla f(x)^\top (y-x) + m/2\,\|y-x\|^2 \\ &\geq f(x) + \nabla f(x)^\top (\tilde{y}-x) + m/2\,\|\tilde{y}-x\|^2 \\ & f(x) - \frac{1}{2m}\|\nabla f(x)\|^2 \end{aligned}$$

Setting as y the minimizer of f, we get

$$f^* \geq f(x) - \frac{1}{2m} \|\nabla f(x)\|^2 \tag{4.1}$$

This implies that if the gradient is small, the point is close to the optimal. We can derive a stopping criterion for the Gradient method of the form

$$\|\nabla f(x)\| \leq (2m\varepsilon)^{1/2} \Rightarrow f(x) - f^* \leq \varepsilon.$$

In this section we assumed that the function is strongly convex, which by Proposition 4.5.2 has a lower bound of m on the smallest eigenvalue of the Hessian of f. To analyze the Gradient descent method, we also assume an upper bound on the maximum eigenvalue of the Hessian of f of the form

$$\nabla^2 f(x) \preceq MI, \tag{4.2}$$

which, in some literature, can be equivalently stated as the requirement on the sub-level sets of f to be closed.

We are ready to present the Gradient descent method.

Algorithm 4: Gradient descent method

Input: starting point $x^{(0)} \in U$
Output: point close to optimal

$k \leftarrow 0$
repeat
  choose a descent direction $\Delta x^{(k)} = -\nabla f(x^{(k)})$
  choose a descent step $t^{(k)}$ with line or backtracking line search
  $x^{(k+1)} \leftarrow x^{(k)} + t^{(k)} \Delta x^{(k)}$
  $k \leftarrow k+1$
until *stopping criteria is met*;
return $x^{(k)}$

In the reminder of this section we perform a convergence analysis of Algorithm 4 for both exact and backtracking line search. Whenever possible, and clear from the context, we use a simpler notation $x^+ = x + t\Delta x$ for $x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}_{GD}$.

As mentioned above, in this analysis we use the assumption that the function is twice continuously differentiable and is strongly convex with a constant m. We need this assumption to exploit properties in Proposition 4.5.2, which combined with Equation (4.2) gives

$$mI \preceq \nabla^2 f(x) \preceq MI.$$

Moreover, by Equation (4.1) we know that:

$$f^* \geq f(x) - \frac{1}{2m} \|\nabla f(x)\|^2.$$

Finally, let $\tilde{f} : \mathbb{R} \to \mathbb{R}$ be a function describing f as a function of the step length in the negative gradient direction Δx_{GD} defined by $\tilde{f}(t) := f(x - t\nabla f(x))$. By Theorem 9.2.8 for $y = x - t\nabla f(x)$ we get that

$$\tilde{f}(t) \leq f(x) - t\|\nabla f(x)\|^2 + \frac{Mt^2}{2} \|\nabla f(x)\|^2. \tag{4.3}$$

Convergence analysis for exact line search

Theorem 4.5.3 Let f be a strongly convex function that satisfies $mI \preceq \nabla^2 f(x) \preceq MI$. For every $\varepsilon > 0$, the Gradient descent method with an exact line search achieves $f(x^{(k)}) - f^* \leq \varepsilon$ for

$$k \geq \frac{\log\left(\left(f(x^{(0)}) - f^*\right)/\varepsilon\right)}{\log(1/c)},$$

for $c = 1 - m/M$.

Proof. At any time step, we use an exact line search that chooses an optimal step size t^*. Thus we can optimize both sides of Equation (4.3) to get

$$f(x^+) = \tilde{f}(t^*) \leq f(x) - \frac{1}{2M}\|\nabla f(x)\|^2,$$

where the right-hand side of Equation (4.3) is a quadratic function attaining its minimum at $t = 1/M$. Subtracting f^* on both sides an applying Equation 4.1 in the form $-\|\nabla f(x)\|^2 \leq 2m(f^* - f(x))$ we get

$$f(x^+) - f^* \leq f(x) - f^* + \frac{m}{M}(f^* - f(x)) = \left(1 - \frac{m}{M}\right)(f(x) - f^*).$$

After k recursive steps, for $c = 1 - m/M$, we get that

$$f(x^{(k)}) - f^* \leq c^k (f(x) - f^*).$$

In particular we get $f(x^{(k)}) - f^* \leq \varepsilon$ after at most

$$\frac{\log\left(\left(f(x^0) - f^*\right)/\varepsilon\right)}{\log(1/c)}$$

steps. ■

To interpret Theorem 4.5.3 think of a numerator in the number of steps as a log of a ratio of the initial suboptimality and the final suboptimality ε. The denominator is fixed for a function f which indicates that constants in the strong convexity affect the convergence speed of the Gradient descent method. We call such a convergence speed linear convergence, since th error lies below a line on a log-linear plot of error versus iteration number.

Convergence analysis for backtracking line search

We start with the following proposition.

Proposition 4.5.4 Let f be a strongly convex function that satisfies $mI \preceq \nabla^2 f(x) \preceq MI$. A backtracking line search algorithms exit condition is satisfied for every $t \in [0, 1/M]$

Proof. We want to show that $\tilde{f}(t) \leq f(x) - \alpha t \|\nabla f(x)\|^2$ for every $t \in [0, 1/M]$. First note that

$$t \in [0, 1/M] \Leftrightarrow t(t - 1/M) \leq 0 \Leftrightarrow -t + \frac{Mt^2}{2} \leq -\frac{t}{2}.$$

By plugging in into Equation (4.3) we get

$$\tilde{f}(t) \leq f(x) - \frac{t}{2}\|\nabla f(x)\|^2 \leq f(x) - \alpha t \|\nabla f(x)\|^2$$

since $\alpha \leq 1/2$. ■

Theorem 4.5.5 Let f be a strongly convex function that satisfies $mI \preceq \nabla^2 f(x) \preceq MI$. For every $\varepsilon > 0$, the Gradient descent method with a backtracking line search achieves $f(x^{(k)}) - f^* \leq \varepsilon$ for

$$k \geq \frac{\log\left(\left(f(x^{(0)}) - f^*\right)/\varepsilon\right)}{\log(1/c)},$$

for $c = 1 - \min\{2m\alpha, 2\beta\alpha m/M\}$.

Proof. By Proposition 4.5.4, a backtracking algorithm exits with $t \geq \beta/M$ or $t = 1$, which gives a bound of

$$f(x^+) \leq f(x) - \min\{\alpha, \alpha\beta/M\} \|\nabla f(x)\|^2.$$

Following exactly the same steps as in Theorem 4.5.3, i.e., subtracting f^* and using the bound $-\|\nabla f(x)\|^2 \leq 2m(f^* - f(x))$ we get

$$f(x^+) - f^* \leq f(x) - f^* + 2m\min\{\alpha, \alpha\beta/M\}(f^* - f(x)) = (1 - \min\{2\alpha m, 2\alpha\beta n/M\})(f(x) - f^*).$$

After k recursive steps, for $c = 1 - \min\{2\alpha m, 2\alpha\beta n/M\}$, we get that

$$f(x^{(k)}) - f^* \leq c^k (f(x) - f^*).$$

In particular we get $f(x^{(k)}) - f^* \leq \varepsilon$ after at most

$$\frac{\log\left(\left(f(x^0) - f^*\right)/\varepsilon\right)}{\log(1/c)}$$

steps. ■

4.6 Newton method

In this section, we analyze Newton's method. We assume that the function is twice continuously differentiable and is strongly convex with a constant m, which combined with Equation (4.2) gives

$$mI \preceq \nabla^2 f(x) \preceq MI.$$

Moreover, we assume that the Hessian of f is Lipschitz continuous on U with constant L, that is

$$\left\|\nabla^2 f(x) - \nabla^2 f(y)\right\| \leq L\|x - y\|.$$

We also introduce *the Newton decrement* defined as

$$\lambda(x) = \left(\nabla f(x)^\top \nabla^2 f(x)^{-1} \nabla f(x)\right)^{1/2}, \tag{4.4}$$

or

$$\lambda(x)^2 = -\nabla f(x)^\top \Delta x_N, \tag{4.5}$$

which will appear in the proofs of convergence for Newton's method. We are ready to present Newton's method.

To prove the convergence of the Newton's method we need the following technical result.

Algorithm 5: Newton's method

Input: starting point $x^{(0)} \in U$, tolerance $\varepsilon > 0$
Output: point x s.t. $f(x) - f^* \leq \varepsilon$

$k \leftarrow 0$
repeat
  choose a descent direction $\Delta x_N^{(k)} \leftarrow -\nabla f(x^{(k)})/\nabla^2 f(x^{(k)})$
  $\lambda^2 \leftarrow \nabla f(x)^\top \nabla^2 f(x)^{-1} \nabla f(x)$
  if $\lambda^2/2 \leq \varepsilon$ **then**
    quit
  end
  choose a descent step $t^{(k)}$ with backtracking line search
  $x^{(k+1)} \leftarrow x^{(k)} + t^{(k)} \Delta x^{(k)}$
  $k \leftarrow k+1$
end;
return $x^{(k)}$

Theorem 4.6.1 Let f be a strongly convex function that satisfies $mI \preceq \nabla^2 f(x) \preceq MI$, whose Hessian is L Lipschitz continuous. Then there exists numbers $0 < \eta \leq m^2/L$ and $\gamma > 0$ such that in the Newton's method for all $k \in \mathbb{N}$ it holds

1. If $\left\|\nabla f(x^{(k)})\right\| \geq \eta$, then
$$f(x^{(k+1)}) - f(x^{(k)}) \leq -\gamma.$$
2. If $\left\|\nabla f(x^{(k)})\right\| < \eta$, then the backtracking algorithm selects $t^{(k)} = 1$ and
$$\frac{L}{2m^2}\left\|\nabla f(x^{(k+1)})\right\| \leq \left(\frac{L}{2m^2}\left\|\nabla f(x^{(k)})\right\|\right)^2.$$

Proof. **First condition:** First, we apply Equation 4.3 (that holds by Theorem 9.2.8 for $y = x + t\Delta x_N$) to derive a lower bound on the step size selected by the line search algorithm

$$\begin{aligned} f(x+t\Delta x_N) &\leq f(x) + t\nabla f(x)^\top \Delta x_N + \frac{Mt^2}{2}\|\Delta x_N\|^2 \\ &\leq f(x) - t\lambda(x)^2 + \frac{Mt^2}{2m}\lambda(x)^2, \end{aligned}$$

where the last inequality follows from

$$\lambda(x)^2 = \Delta x_N^\top \nabla^2 f(x) \Delta x_N \geq m\|\Delta x_N\|^2.$$

Now we can check that the step size $\widehat{t} = m/M$ satisfies the exit condition of the line search since $\alpha \leq 1/2$ thus

$$f(x+\widehat{t}\Delta x_N) \leq f(x) - \frac{m}{2M}\lambda(x)^2 \leq f(x) - \alpha\widehat{t}\lambda(x)^2.$$

Therefore the backtracking line search returns the step size $t \geq \beta m/M$, resulting in a decrease of

the objective function

$$\begin{aligned} f\left(x^{+}\right)-f(x)- &\leq \alpha t \lambda(x)^2 \\ &\leq -\alpha\beta\frac{m}{M}\lambda(x)^2 \\ &\leq -\alpha\beta\frac{m}{M^2}\left\|\nabla f(x)\right\|^2 \\ &\leq -\alpha\beta\eta^2\frac{m}{M^2}, \end{aligned}$$

where the second last inequality follows since

$$\lambda(x)^2 = \nabla f(x)^{\top}\left(\nabla^2 f(x)\right)^{-1}\nabla f(x) \geq 1/M\left\|\nabla f(x))\right\|^2.$$

So the claim holds for $\gamma = \alpha\beta\eta^2\frac{m}{M^2}$.

Second condition: First, we assume that the backtracking line search selects unit time steps, and we prove it at the end.

Applying the Lipschitz condition we get

$$\begin{aligned} \left\|\nabla f(x^{+})\right\| &= \left\|\nabla f\left(x+\Delta x_N\right)-\nabla f(x)-\nabla^2 f(x)\Delta x_N\right\| \\ &= \left\|\int_0^1\left(\nabla^2 f(x+t\Delta x_N)-\nabla^2 f(x)\right)\Delta x_N dt\right\| \\ &\leq L/2\left\|\Delta x_N\right\|^2 \\ &= L/2\left\|\nabla^2 f(x)^{-1}\nabla f(x)\right\|^2 \\ &\leq \frac{L}{2m^2}\left\|\nabla f(x)\right\|^2. \end{aligned}$$

Next we want to show, that backtracking line search always picks unit time steps. We will begin with the calculations and calculate the specific η at the end.

Begin by defining $g(t) = f(x+t\Delta x_N)$. Then we have to estimate $g(1)$. The idea is to use the Lipschitz condition along the descent direction to get an estimate for $g''(t)$ and then integrate the inequality twice. First observe

$$\left\|\nabla^2 f(x+t\Delta x_N)-\nabla^2 f(x)\right\| \leq tL\left\|\Delta x_N\right\|. \tag{4.6}$$

Furthermore $g''(t) = \Delta x_N^{\top}(\nabla^2 f(x+t\Delta x_N))\Delta x_N$ which follows by direct calculations. Combining this with (4.6) yields:

$$\mid g''(t)-g''(0)\mid \leq tL\left\|\Delta x_N\right\|^3.$$

Integrating this gives:

$$g'(t)-g'(0) = \int_0^t g''(s)ds \leq \int_0^t g''(0)+sL\left\|\Delta x_N\right\|^3 ds = tg''(0)+\frac{t^2L}{2}\left\|\Delta x_N\right\|^3.$$

Moving $g'(0)$ to the other side and integrating again yields:

$$g(t)-g(0) = \int_0^t g'(s)ds \leq \int_0^t g'(0)+sg''(0)+\frac{s^2}{2}\left\|\Delta x_N\right\|^3 ds = tg'(0)+\frac{t^2}{2}g''(0)+\frac{t^3L}{6}\left\|\Delta x_N\right\|^3.$$

So alltogether we get:

$$g(t) = g(0)+tg'(0)+\frac{t^2}{2}g''(0)+\frac{t^3L}{6}\left\|\Delta x_N\right\|^3.$$

We are interested in $f(x+\Delta x_N) = g(1)$ so we plug in $t = 1$ above.

$$g(1) = g(0) + g'(0) + \frac{1}{2}g''(0) + \frac{L}{6}\|\Delta x_N\|^3.$$

Now we have to bound $\|\Delta x_N\|^3$, first using λ. For this we use (4.5), together with the definition of Δx_n

$$\lambda(x)^2 = -\nabla f(x)^\top \Delta x_N = \Delta x_N^\top \nabla^2 f(x) \Delta x_N.$$

We can bound the final term by using strict convexity, $mI \preceq \nabla^2 f(x)$, so $\nabla^2 f(x) - mI$ is positive semidefinite. Hence $\Delta x_N^\top \left(\nabla^2 f(x) - mI\right) \Delta x_N \geq 0$ and

$$\lambda(x)^2 \geq \Delta x_N^\top mI \Delta x_N = m\|x\|^2.$$

We are left with calculating $g(0), g'(0), g''(0)$. For this observe:

$$\begin{aligned}
g(0) &= f(x + 0 \cdot \Delta x_N) = f(x), \\
g'(0) &= \nabla f(x)^\top \Delta x_N \overset{(4.5)}{=} -\lambda(x)^2, \\
g''(0) &= \Delta x_N^\top (\nabla^2 f(x)) \Delta x_N \overset{(4.4)}{=} \lambda(x)^2.
\end{aligned}$$

So collecting everything we get

$$g(1) = f(x) - \frac{1}{2}\lambda(x)^2 + \frac{L\lambda(x)^3}{6m^{3/2}}.$$

To show that backtracking line search returns unit step size we need that

$$\begin{aligned}
f(x) - \frac{1}{2}\lambda(x)^2 + \frac{L\lambda(x)^3}{6m^{3/2}} = g(1) = f(x+\Delta x_N) &\leq f(x) + \alpha \nabla f(x)^\top \Delta x_N = f(x) - \alpha\lambda(x)^2 \\
\frac{\lambda(x)^3}{6m^{3/2}} &\leq \left(\frac{1}{2} - \alpha\right)\lambda(x)^2 \\
\lambda &\leq \frac{3(1-2\alpha)m^{3/2}}{L} \qquad (4.7)
\end{aligned}$$

whenever $\|f(x)\| \leq \eta$.

So we need to connect $\lambda(x)$ with η. For this we use (4.4) and again positive semidefiniteness of $\nabla^2 f(x) - mI$.

$$\lambda(x) = \left(\nabla f(x)^\top \nabla^2 f(x)^{-1} \nabla f(x)\right)^{1/2} \leq \left(\nabla f(x)^\top \frac{1}{m} I \nabla f(x)\right)^{1/2} = m^{-1/2}\|f(x)\| = m^{-1/2}\eta$$

Plugging this into the backtracking condition (4.7) and solving for η gives that the condition is surely fulfilled (using $\alpha < 0.5$) if:

$$\eta \leq \frac{3(1-2\alpha)m^2}{L}$$

which finally gives us the threshold we wanted.

■

The two conditions in Theorem 4.6.1 define two phases of Newton's method. The first phase is called the *damped Newton's phase*. The second phase is called the *quadratically convergent phase* since the convergence in this phase is very rapid, it is double exponential. The name comes from the fact that the error is a quadratic function on a log-linear plot of error versus iteration number. The second phase is also sometimes called the *pure Newton's phase* since, in this phase, the line search always chooses a step size $t = 1$.

As we could see in the proof of Theorem 4.6.1 once Newton's method hits the pure Newton's phase, it never goes back to the damped Newton's phase again. Since the convergence in the pure Newton's phase is quadratic, in practice, the goal is to reach the second phase of Newton's method as soon as possible. We will see similar behavior in the next chapter, where we study algorithms for constrained convex optimization, namely, the interior point method, where staying in the quadratic convergence region will be a key aspect of obtaining the algorithm's running time guarantees.

We are ready to prove the convergence of Newton's method.

Theorem 4.6.2 Let f be s strongly convex function that satisfies $mI \preceq \nabla^2 f(x) \preceq MI$, whose Hessian is L Lipschitz continuous. For every $\varepsilon > 0$, the Newton's method with a backtracking line search achieves $f(x^{(k)}) - f^* \leq \varepsilon$ for

$$k \geq \frac{f(x^{(0)}) - f^*}{\gamma} + \log\log\frac{2m^3}{L\varepsilon},$$

for γ being a constant from Theorem 4.6.1.

Proof. First, we note that once Newton's method reaches a quadratic convergence, it holds for all the following iterations of the algorithm. Indeed, let k^* be the first moment in which the second condition of Theorem 4.6.1 is satisfied, then we have

$$\frac{L}{2m^2}\left\|\nabla f(x^{(k^*+1)})\right\| \leq \left(\frac{L}{2m^2}\left\|\nabla f(x^{(k^*)})\right\|\right)^2,$$

which after applying recursively for any $\ell \geq k^*$ we get

$$\frac{L}{2m^2}\left\|\nabla f(x^{(\ell)})\right\| \leq \left(\frac{L}{2m^2}\left\|\nabla f(x^{(k^*)})\right\|\right)^{2^{l-k^*}} \leq \left(\frac{1}{2}\right)^{2^{l-k^*}},$$

where the last inequality holds since $\left\|\nabla f(x^{(k^*)})\right\| < \eta \leq m^2/L$.

Thus, by Equation (4.1) (which follows from Proposition 4.5.2 and is not only applicable for the Gradient descent algorithm) we get

$$f\left(x^{(\ell)}\right) - f^* \leq \frac{1}{2m}\left\|\nabla f\left(x^{(\ell)}\right)\right\|^2 \leq \frac{2m^3}{L^2}\left(\frac{1}{2}\right)^{2^{l-k+1}}.$$

This implies that Newton's method first runs the damped Newton phase for at most k^* steps upper bounded by

$$\frac{f\left(x^{(0)}\right) - f^*}{\gamma}$$

and then pure Newton phase for $k - k^*$ steps upper bounded by

$$\log\log\frac{2m^3}{L\varepsilon}.$$

■

4.7 Self-concordant functions

In previous Sections we studied different algorithms to solve unconstrained convex optimization problems. For all the algorithm presented above the convergence depends on some constants characterizing the function to be optimized. As a consequence, since the constants are rarely known in practice, the complexity of the algorithms remains unknown. For this reason we are interested in setting, and an algorithm, for which the complexity of does not depend of the constants characterizing the objective function. In this section we study self-concordant functions and show convergence of the Newton's method that does not involve any, possibly unknown, constants. In other word that self-concordance can replace the two requirements from the previous chapter, namely, $mI \preceq \nabla^2 f(x) \prec MI$ and $\left\|\nabla^2 f(x) - \nabla^2 f(y)\right\| \leq L\|x-y\|$.

We start with a definition of self-concordant function in one dimension.

Definition 4.7.1 Let $f: U \subseteq \mathbb{R} \to \mathbb{R}$ be a convex, three times differentiable function. The function f is called *self-concordant* if it satisfies

$$\left|f'''(x)\right| \leq 2f''(x)^{3/2},$$

for all $x \in U$.

The first observation we can make is that the constant 2 in front of the expression on the rand-hand side in Definition 4.7.1 is for convenience only. As the next proposition shows, a different constant can lead to self-concordant function after appropriate scaling.

Proposition 4.7.2 Let $f: U \subseteq \mathbb{R} \to \mathbb{R}$ be a convex, three times differentiable function. If, for some $k > 0$ the function f satisfies

$$\left|f'''(x)\right| \leq kf''(x)^{3/2},$$

for all $x \in U$, then the function $\tilde{f}(x) := k^2/4 f(x)$ is self-concordant.

Proof. Indeed, note that for every $x \in U$ we have

$$\begin{aligned}
\left|\tilde{f}'''(x)\right| &= \frac{k^2}{4}\left|f'''(x)\right| \\
&\leq \frac{k^3}{4} f''(x)^{3/2} \\
&= \frac{k^3}{4}\left(\frac{4}{k^2}\tilde{f}''(x)\right)^{3/2} \\
&= 2\tilde{f}''(x)^{3/2}.
\end{aligned}$$

■

The second observation is that invariant under affine transformation of variables.

Proposition 4.7.3 Let $f: \mathbb{R} \to \mathbb{R}$ be a self-concordant function. Then, for any $a \in \mathbb{R}$, $a \neq 0$ and any $b \in \mathbb{R}$ the function $\tilde{f}(y) := f(ay+b)$ is also self-concordant.

Proof. Let $x = ay + b$. An easy application of the chain rule for higher derivatives gives

$$\tilde{f}''(y) = a^2 f''(x) \qquad \text{and} \qquad \tilde{f}'''(y) = a^3 f'''(x).$$

By the self-concordance of f we get

$$\left|\tilde{f}'''(y)\right| = \left|a^3 f'''(x)\right| \leq 2\left|a^3\right| f''(x)^{3/2} = 2\left(a^2 f''(x)\right)^{3/2} = 2\tilde{f}(y)^{3/2}.$$

■

In the following we present several examples of well-known functions that are self-concordant.

■ **Example 4.7.4** The following functions are self-concordant:

- Negative logarithm:
$$f(x) = -\log x.$$
- Negative entropy and negative logarithm:
$$f(x) = x\log x - \log x,$$

■

We are ready to extend the definition of the self-concordant function to higher dimensions.

Definition 4.7.5 Let $f : U \subseteq \mathbb{R}^n \to \mathbb{R}$ be a convex, three times differentiable function. The function f is called *self-concordant* if for every $v \in \mathbb{R}^n$ the function $\tilde{f}(t) : \mathbb{R} \to \mathbb{R}$ defined as $\tilde{f}(t) := f(x+tv)$ is self-concordant for all $x \in U$.

In the following proposition we summarize the properties of multidimensional self-concordant functions.

Proposition 4.7.6 Let $f : \mathbb{R}^n \to \mathbb{R}$ be a self-concordant function. Then the following properties hold

1. **scaling:** For every $a \geq 1$, the function af is self-concordant.
2. **summing:** For every self-concordant function $g : \mathbb{R}^n \to \mathbb{R}$, $f+g$ is self-concordant.
3. **affine composition:** For every $A \in \mathbb{R}^{n\times m}$, and $b \in \mathbb{R}^n$, the function $f(Ax+b)$ is self-concordant.

Proof. In Exercise 4.5. ■

4.7.1 Newton's method for self-concordant functions

In this subsection we analyze convergence of Newton's method for self-concordant functions. We will heavily use the following properties of self-concordant functions.

Proposition 4.7.7 Let $f : U \subseteq \mathbb{R} \to \mathbb{R}$ be a strongly convex self-concordant function. Then the following bounds hold

$$\frac{f''(0)}{\left(1+tf''(0)^{1/2}\right)^2} \leq f''(t) \leq \frac{f''(0)}{\left(1-tf''(0)^{1/2}\right)^2},$$

where the lower bound holds for all $t \in U$ and the upper bound holds for $t \in U$ s.t. $0 \leq t \leq f''(0)^{1/2}$.

Proof. Exercise 4.6. ■

Moreover, for strongly convex, self-concordant functions, the distance between the value of the function at any point and the optimal value can be measured using the Netwon's decrement. This serves as an alternative measure to Equation (4.1).

Proposition 4.7.8 Let $f : U \subseteq \mathbb{R} \to \mathbb{R}$ be a strongly convex, self-concordant function. Then, for every $\varepsilon > 0$ the following bounds holds

$$f^* - f(x) \geq -\lambda(x)^2.$$

Proof. Exercise 4.7 ■

Finally, we will use the following technical result

Proposition 4.7.9 Let $f: U \subseteq \mathbb{R} \to \mathbb{R}$ be a strongly convex, self-concordant function. If $\lambda(x) < 1$, then $\lambda(x+\Delta x_N) \leq \lambda(x)^2/(1-\lambda(x))^2$.

Proof. Exercise 4.8 ■

To prove the convergence of the Newton's method we need the following technical result, that can be seen as an analogue of Theorem 4.6.1. Two main difference between those theorems are: first we do not have constants m, M and L, second the norm of the gradient of f was replaced by the Newton's decrement. The reason behind the second change is that, as presented in Proposition 4.7.8, one can use Newton's decrement to measure the distance between the value of the function at step k and the optimal value of the function in a similar way as with the norm of the gradient, as in Equation 4.1.

Theorem 4.7.10 Let f be a strongly convex self-concordant function. Then there exists numbers $0 < \eta \leq 1/4$ and $\gamma > 0$ such that in the Newton's method for all $k \in \mathbb{N}$ it holds

1. If $\lambda\left(x^{(k)}\right) > \eta$, then
$$f(x^{(k+1)}) - f(x^{(k)}) \leq -\gamma.$$
2. If $\lambda\left(x^{(k)}\right) \leq \eta$, then the backtracking algorithm selects $t^{(k)} = 1$ and
$$2\lambda\left(x^{(k+1)}\right) \leq \left(2\lambda\left(x^{(k)}\right)\right)^2.$$

Proof. **First condition:** Let $\tilde{f}(t) = f(x+t\Delta x_N)$. First we note that

$$f'(t) = f'(x+t\Delta x_N)\Delta x_N,$$

thus

$$f'(0) = \nabla f(x)\Delta x_N = -\lambda(x)^2$$

and

$$f''(t) = f''(x+t\Delta x_N)(\Delta x_N)^2 = \nabla^2 f(x)\frac{-\nabla f(x)}{\nabla^2 f(x)}\Delta x_N,$$

thus

$$f''(0) = -\nabla f(x)\Delta x_N = \lambda(x)^2.$$

Next we use an upper bound in Proposition 4.7.6, that after integrating twice for $0 \leq t < 1/\lambda(x)$ yields

$$\begin{aligned}\tilde{f}(t) &\leq \tilde{f}(0) + t\tilde{f}'(0) - t\tilde{f}''(0)^{1/2} - \log\left(1 - t\tilde{f}''(0)^{1/2}\right) \\ &= \tilde{f}(0) - t\lambda(x)^2 - t\lambda(x) - \log(1 - t\lambda(x)) \end{aligned} \tag{4.8}$$

Now we can check that the step size $\widehat{t} = 1/(1+\lambda(x))$ satisfies the exit condition of the line search

$$\begin{aligned}\tilde{f}(x) &\leq \tilde{f}(0) - \widehat{t}\lambda(x)^2 - \widehat{t}\lambda(x) - \log\left(1 - \widehat{t}\lambda(x)\right) \\ &= \tilde{f}(0) - \lambda(x) + \log(1+\lambda(x)) \\ &\leq \tilde{f}(0) - \alpha\frac{\lambda(x)^2}{1+\lambda(x)} \\ &= \tilde{f}(0) - \alpha\lambda(x)^2\widehat{t},\end{aligned}$$

where the second inequality follows from the fact that for $x \geq 0$ we have

$$-x + \log(1+x) + \frac{x^2}{2(1+x)} \leq 0$$

Therefore the backtracking line search returns the step size $t \geq \beta / (1+\lambda(x))$, resulting in a decrease of the objective function

$$\tilde{f}(t) - \tilde{f}(0) = f(x + t\Delta x_N) - f(x) \leq -\alpha\beta \frac{\lambda(x)^2}{1+\lambda(x)}.$$

So the claim holds for $\gamma = \alpha\beta \frac{\eta^2}{1+\eta}$.

Second condition: First we show that the backtracking line search algorithm always takes unit step. Let $\eta = (1-2\alpha)/4$ (which, for $0 \leq \alpha \leq 1/2$ satisfies $\eta \leq 1/4$). We want to show that if $\lambda(x^{(k)}) \leq (1-2\alpha)/4$, then the backtracking line search accepts unit time step.

We note that the upper bound in Equation (4.8) implies that the unit step is accepted by the backtracking line search. Indeed, note that once the algorithm enters quadratically convergence phase at step k^*, for every $k > k^*$ we have $\lambda(x^{(k)} \leq (1-2\alpha)/2$, since in every step $\lambda(x)$ halves, thus

$$\begin{aligned}
\tilde{f}(1) &\leq \tilde{f}(0) - \lambda(x)^2 - \lambda(x) - \log(1-\lambda(x)) \\
&\leq \tilde{f}(0) - \frac{1}{2}\lambda(x)^2 + \lambda(x)^3 \\
&\leq \tilde{f}(0) - \alpha\lambda(x)^2,
\end{aligned}$$

where the second inequality follows from the fact that $0 \leq x \leq 0.81$ we have $-x - \log(1-x) \leq \frac{1}{2}x^2 + x^3$. The second part of the claim follows by Proposition 4.7.9 for $\lambda(x) \leq 1/4$ which implies that

$$\lambda(x^+) \leq 2\lambda(x)^2$$

for $\lambda(x^+) \leq \eta$. ■

By Theorem 4.7.10, analogous to the proof of Theorem 4.6.1, once Newton's method hits the pure Newton's phase, it never goes back to the damped Newton's phase again. We are ready to prove the convergence of Newton's method for self-concordant functions.

Theorem 4.7.11 Let f be s strongly convex self-concordant function. For every $\varepsilon > 0$, the Newton's method with a backtracking line search achieves $f(x^{(k)}) - f^* \leq \varepsilon$ for

$$k \geq \frac{f(x^{(0)}) - f^*}{\gamma} + \log\log\frac{1}{\varepsilon},$$

for γ being a constant from Theorem 4.7.10.

Proof. First, we note that once Newton's method reaches a quadratic convergence, it holds for all the following iterations of the algorithm. Indeed, let k^* be the first moment in which the second condition of Theorem 4.7.10 is satisfied, then we have

$$2\lambda\left(x^{(k^*+1)}\right) \leq \left(2\lambda\left(x^{(k^*)}\right)\right)^2.$$

which after applying recursively for any $\ell \geq k^*$ we get

$$2\lambda\left(x^{(l)}\right) \leq \left(2\lambda\left(x^{(k^*)}\right)\right)^{2^{l-k^*}} \leq (2\eta)^{2^{l-k^*}} \leq \left(\frac{1}{2}\right)^{2^{l-k^*}}$$

where the last two inequalities holds since $\lambda\left(x^{(k^*)}\right) \leq \eta \leq 1/4$.

Thus, by Proposition 4.7.8 we get

$$f\left(x^{(\ell)}\right) - f^* \leq \lambda\left(x^{(l)}\right)^2 \leq \frac{1}{4}\left(\frac{1}{2}\right)^{2^{l-k^*+1}} \leq \left(\frac{1}{2}\right)^{2^{l-k^*+1}}.$$

This implies that Newton's method first runs the damped Newton phase for at most k^* steps upper bounded by

$$\frac{f\left(x^{(0)}\right) - f^*}{\gamma}$$

and then pure Newton phase for $k - k^*$ steps upper bounded by

$$\log\log\frac{1}{\varepsilon}.$$

■

4.8 Exercises

Exercise 4.1 — Programming Exercise. Implement gradient descent with backtracking line search as described in Algorithm 4. You can choose the stopping criterion yourself. Try different functions as well as different values for α and β. On the webpage you can find a notebook with further information.

Exercise 4.2 In this exercise we want to derive under which conditions we can say something about the convergence of a general descent algorithm for continuous differentiable f. Therefore we look at general algorithms of the form as introduced in Algorithm 1. Our goal will be to find

Algorithm 6: Descent method

Input : Starting point $x^{(0)} \in \mathbb{R}^n$
output : A sequence of iterations $\left(x^{(k)}\right) \subseteq \mathbb{R}^n$

$k \leftarrow 0$
repeat
  Choose a descent direction $\Delta x^{(k)}$, i.e. $\nabla f\left(x^{(k)}\right)^\top \Delta x^{(k)} < 0$
  Choose a descent step $t^{(k)}$ such that $f\left(x^{(k)} + t^{(k)}\Delta x^{(k)}\right) < f\left(x^{(k)}\right)$
  $x^{(k+1)} \leftarrow x^{(k)} + t^{(k)}\Delta x^{(k)}$
  $k \leftarrow k+1$
until $\nabla f\left(x^{(k)}\right) = 0$;
return $\left(x^{(k)}\right)$

stationary points of f which in some cases (e.g. when f is convex) are also minimas. It is clear that this is achieved when Algorithm 6 stops after a finite number of iterations. So we will suppose that $\left(x^{(k)}\right)_{k\in\mathbb{N}}, \left(\Delta x^{(k)}\right)_{k\in\mathbb{N}}, \left(t^{(k)}\right)_{k\in\mathbb{N}}$ are infinite sequences. In this case we call a subsequence $\left(t^{(k)}\right)_{k\in K}$ of stepsizes *admissible*, if

$$i)\ \forall k \in \mathbb{N}\colon f\left(x^{(k)} + t^{(k)}\Delta x^{(k)}\right) \leq f\left(x^{(k)}\right),$$

$$ii)\ f\left(x^{(k)} + t^{(k)}\Delta x^{(k)}\right) - f\left(x^{(k)}\right) \to 0 \ \Rightarrow\ \left(\frac{\nabla f\left(x^{(k)}\right)^\top \Delta x^{(k)}}{\left\|\Delta x^{(k)}\right\|}\right)_{k\in K} \to 0.$$

A subsequence $\left(\Delta x^{(k)}\right)_{k\in K}$ of descent directions is called *admissible*, if

$$i)\ \forall k \in \mathbb{N}\colon \nabla f\left(x^{(k)}\right)^\top \Delta x^{(k)} < 0,$$

$$ii)\ \left(\frac{\nabla f\left(x^{(k)}\right)^\top \Delta x^{(k)}}{\left\|\Delta x^{(k)}\right\|}\right)_{t\in K} \to 0 \ \Rightarrow\ \left(\nabla f\left(x^{(k)}\right)\right)_{t\in K} \to 0.$$

Prove to following statement.

a) Suppose $\bar{x}$ is a limit point of $\left(x^{(k)}\right)$, i.e. there exists a subsequence $\left(x^{(k)}\right)_{k\in K} \to \bar{x}$. Furthermore suppose that the subsequences $\left(t^{(k)}\right)_{k\in K}, \left(\Delta x^{(k)}\right)_{k\in K}$ indexed by the same index set are admissible. Then $\bar{x}$ is a stationary point.

Now we want to find rules which imply admissibility. Therefore we say $\left(\Delta x^{(k)}\right)_{k\in K}$ fulfils an *angle inequality*, if there exists a $\eta \in (0,1)$ such that

$$\cos\angle\left(-\nabla f\left(x^{(k)}\right), \Delta x^{(k)}\right) = \frac{-\nabla f\left(x^{(k)}\right)^\top \Delta x^{(k)}}{\left\|\nabla f\left(x^{(k)}\right)\right\| \left\|\Delta x^{(k)}\right\|} \geq \eta$$

for all $k \in K$.

b) Prove that $\left(\Delta x^{(k)}\right)_{k\in K}$ is admissible if it is generated by Algorithm 6 and fulfils the angle inequality.

Finally we call stepsizes $\left(t^{(k)}\right)_{k\in K}$ *efficient*, if there exists a $\theta > 0$ such that

$$f\left(x^{(k)} + t^{(k)}\Delta x^{(k)}\right) \leq f\left(x^{(k)}\right) - \theta\left(\frac{\nabla f\left(x^{(k)}\right)^{\top}\Delta x^{(k)}}{\left\|\Delta x^{(k)}\right\|}\right)^2$$

for all $k \in K$.

c) Prove that $\left(t^{(k)}\right)_{k\in K}$ is admissible if it is generated by Algorithm 6 and efficient.

Exercise 4.3 — Different types of Convexity. Let $\emptyset \neq X \subseteq \mathbb{R}^n$ be a closed convex set and $f\colon X \to \mathbb{R}$ continuous. We look at the minimization problem

$$\min_{x\in X} f(x) \tag{P}$$

and want to decide whether it has a global solution or not. Provide a proof or counterexample for the following statements.

i) If f is convex, (P) has a (not necessarily unique) global minimizer.
ii) If f is strictly convex, (P) has a (not necessarily unique) global minimizer.
iii) If f is strongly convex with constant $m > 0$, then (P) has a unique global solution.

Exercise 4.4 Prove that function in Example 4.7.4 are self-concordant.

Exercise 4.5 Prove Proposition 4.7.6.

Exercise 4.6 — Proposition 4.7.7. Let $0 \in U \subseteq \mathbb{R}$ be an open interval and $f\colon U \to \mathbb{R}$ a strictly convex and self-concordant function. Prove that for all nonnegative $t \in U$

$$\frac{f''(0)}{\left(1 + t f''(0)^{1/2}\right)^2} \leq f''(t).$$

Further prove

$$f''(t) \leq \frac{f''(0)}{\left(1 - t f''(0)^{1/2}\right)^2}$$

for $t \in U$ with $0 \leq t \leq f''(0)^{1/2}$.

Exercise 4.7 — Proposition 4.7.8. Let $0 \in U \subseteq \mathbb{R}^n$ be an open convex set and $f\colon U \to \mathbb{R}$ a strictly convex and self-concordant function. Prove that

$$f(x) - f* \leq \lambda(x)^2$$

if $\lambda(x) \leq 0.68$.

Exercise 4.8 — Proposition 4.7.9. Let $U \subseteq \mathbb{R}^n$ be an open convex set and $f\colon U \to \mathbb{R}$ a strictly convex and self-concordant function. Let $x \in U$ and suppose $\lambda(x) \leq 1$. Prove

$$\lambda(x^+) \leq \frac{\lambda(x)^2}{(1-\lambda(x))^2},$$

where $x^+ \coloneqq x - \nabla^2 f(x)^{-1}\nabla f(x)$.
Hint: You may use that

$$(1-t\alpha)^2\nabla^2 f(x) \preceq \nabla^2 f(x+tv) \preceq \frac{1}{(1-t\alpha)^2}\nabla^2 f(x),$$

where $v \in \mathbb{R}^n$,

$$\alpha \coloneqq \left(v^\top \nabla^2 f(x) v\right)^{1/2}$$

and $t \in [0,\alpha)$ such that $(x+tv) \in U$.

4.9 Solutions

Solution of the Exercise 4.2

For *a*), we will first prove that $f\left(x^{(k)}\right) \to f(\bar{x})$. Therefore note that by the choice of $t^{(k)}$ we get that $f\left(x^{(k)}\right)$ is a strictly monotonically decreasing function. Thus $f\left(x^{(k)}\right)$ has a limit $\bar{f} \in \mathbb{R} \cup \{-\infty\}$ and hence $\left(f\left(x^{(k)}\right)\right)_{k\in K} \to \bar{f}$. Since f is continuous, we get

$$f(\bar{x}) = \bar{f}$$

and in particular $\bar{f} \neq -\infty$. Therefore we have a sequence in $\mathbb{R}$ with a finite limes, in particular $f\left(x^{(k)}\right)$ is a cauchy sequence and we get

$$f\left(x^{(k)} + t^{(k)}\Delta x^{(k)}\right) - f\left(x^{(k)}\right) = f\left(x^{(k+1)}\right) - f\left(x^{(k)}\right) \to 0.$$

Therefore the admissibility of $\left(t^{(k)}\right)_{k\in K}$ implies

$$\left(\frac{\nabla f\left(x^{(k)}\right)^\top \Delta x^{(k)}}{\left\|\Delta x^{(k)}\right\|}\right)_{k\in K} \to 0.$$

Now the admissibility of $\left(\Delta x^{(k)}\right)_{k\in K}$ shows

$$\left(\nabla f\left(x^{(k)}\right)\right)_{t\in K} \to 0$$

and since f is continuous differentiable, this implies $\nabla f(\bar{x}) = 0$ which shows the claim.

The second claim *b*) follows from rewriting

$$\frac{-\nabla f\left(x^{(k)}\right)^\top \Delta x^{(k)}}{\left\|\nabla f\left(x^{(k)}\right)\right\| \left\|\Delta x^{(k)}\right\|} \geq \eta \Leftrightarrow \frac{-\nabla f\left(x^{(k)}\right)^\top \Delta x^{(k)}}{\left\|\Delta x^{(k)}\right\|} \geq \eta \left\|\nabla f\left(x^{(k)}\right)\right\|$$

and due to the positivity of η and the nonnegativity of norms.

The last claim *c*) again directly follows from the definition of efficiency by rewriting

$$f\left(x^{(k)} + t^{(k)}\Delta x^{(k)}\right) \leq f\left(x^{(k)}\right) - \theta\left(\frac{\nabla f\left(x^{(k)}\right)^\top \Delta x^{(k)}}{\left\|\Delta x^{(k)}\right\|}\right)^2$$

to

$$\theta\left(\frac{\nabla f\left(x^{(k)}\right)^\top \Delta x^{(k)}}{\left\|\Delta x^{(k)}\right\|}\right)^2 \leq f\left(x^{(k)}\right) - f\left(x^{(k)} + t^{(k)}\Delta x^{(k)}\right)$$

and positivity of θ.

Solution of the Exercise 4.3

i,*ii*) We give a counterexample for *ii*) which also can be used for *i*). First note that $\mathbb{R}$ is closed and convex and hence $\exp\colon \mathbb{R} \to \mathbb{R}$ fulfils all requirements. Since $\exp(x) \overset{x\to-\infty}{\longrightarrow} 0$ and $\exp > 0$ we get that the infimum is given by 0 which is not attained.

iii) *Proof:* The proof will consist of 2 parts. We will first show that a continuous coercive function f has a global solution. Here f is called coercive, if

$$\lim_{x\in X, \|x\|\to\infty} f(x) = \infty.$$

Afterwards we will prove that this property holds in *iii*) and proof uniqueness.

For the first part we define the sublevel set

$$N_f(y) := \{ x \in X \mid f(x) \leq f(y) \},$$

where $y \in X$. First note that $N_f(y) = X \setminus f^{-1}((f(y), \infty))$ is closed. Next suppose that $N_f(y)$ is unbounded, i.e. there exists a sequence $\left(x^{(k)}\right) \subseteq N_f(y)$ with $\left\|x^{(k)}\right\| \to \infty$. Since we assume f to be coercive, this is not possible and hence $N_f(y)$ is compact. By definition of the sublevel set we can pick any $y \in X \neq \emptyset$ and get

$$\min_{x \in X} f(x) \Leftrightarrow \min_{x \in N_f(y)} f(x).$$

Now we get a global solution since f is continuous and $N_f(y)$ is compact. This concludes the first step.

For the second step first recall that f is strongly convex with parameter $m > 0$ iff for all $x, y \in X$ and $\lambda \in [0,1]$ we have

$$f(\lambda x + (1-\lambda) y) + \frac{\lambda(1-\lambda)m}{2} \|x-y\|^2 \leq \lambda f(x) + (1-\lambda) f(y).$$

Now suppose that for some fixed $y \in X$ we have $x \in X$ that satisfies $\|x-y\| > 1$. Then we can plug $\lambda = \frac{1}{\|x-y\|}$ into the inequality and get

$$\begin{aligned} f\left(y + \frac{x-y}{\|x-y\|}\right) + \frac{\left(\frac{\|x-y\|-1}{\|x-y\|}\right)m}{2\|x-y\|} \|x-y\|^2 &= f\left(y + \frac{x-y}{\|x-y\|}\right) + (\|x-y\|-1)\frac{m}{2} \\ &\leq \frac{1}{\|x-y\|} f(x) + \left(1 - \frac{1}{\|x-y\|}\right) f(y). \end{aligned}$$

Reordering gives

$$\begin{aligned} f(x) &\geq \|x-y\| f\left(y + \frac{x-y}{\|x-y\|}\right) + f(y) - \|x-y\| f(y) + \frac{m}{2}\left(\|x-y\|^2 - \|x-y\|\right) \\ &= f(y) + \|x-y\| \left(f(z) - f(y) - \frac{m}{2}\right) + \frac{m}{2}\|x-y\|^2, \end{aligned}$$

where $z := y + \frac{x-y}{\|x-y\|}$. This allows us to lower bound $f(x)$ by constants $C_1, C_2 \geq 0$ independent of x via

$$\begin{aligned} C_1 &= |f(y)| \\ C_2 &= \frac{m}{2} + |f(y)| + \sup_{\xi \in X, \|y-\xi\|=1} |f(\xi)| \end{aligned}$$

in the form

$$f(x) \geq -C_1 - C_2 \|x-y\| + \frac{m}{2}\|x-y\|^2.$$

Here C_2 again exists due to continuity and compactness. This proves that f is coercive and hence we get a global solution. Since strong convexity implies strict convexity, we get the uniqueness of this solution.

Solution of the Exercise 4.6

First note that the self-concordance condition $|f'''(x)| \leq 2f''(x)^{3/2}$ can be expressed as

$$\left| \frac{d}{dx} \left(f''(x)^{-1/2} \right) \right| \leq 1. \tag{4.9}$$

if $f''(x) \neq 0$. This follows from the calculation

$$\frac{d}{dx} \left(f''(x)^{-1/2} \right) = -\frac{1}{2} \frac{f'''(x)}{f''(x)^{3/2}}$$

by taking absolute values on both sides. Now suppose $f''(x) = 0$ for some $x \in U$ but $f''(y) > 0$ for a $y \in U$. Then without loss of generality we may assume that $f''(z) > 0$ for all $z \in [y,x)$ (otherwise we take the infimum over all x or change the order in the interval) and hence we can apply (4.9) for such z, i.e.

$$-1 \leq \frac{d}{dz} f''(z)^{-1/2} \leq 1$$

and integrating from y to x gives

$$f''(x)^{-1/2} - f''(y)^{-1/2} \leq x - y < \infty$$

which contradicts $f''(x) = 0$. Therefore we either have that either (4.9) holds on all of U, or $f'' \equiv 0$.

Now we can proof the actual claim. Note that in the case $f'' \equiv 0$ all claims hold trivially, so suppose $f'' > 0$. In this case we can rewrite (4.9) and integrate to get

$$-t \leq \int_0^t \frac{d}{d\tau} \left(f''(\tau)^{-1/2} \right) d\tau \leq t,$$

hence

$$-t \leq f''(t)^{-1/2} - f''(0)^{-1/2} \leq t.$$

Rewriting gives

$$\begin{aligned}
&-t \leq f''(t)^{-1/2} - f''(0)^{-1/2} \leq t \\
&\Leftrightarrow -t + f''(0)^{-1/2} \leq f''(t)^{-1/2} \leq t + f''(0)^{-1/2} \\
&\overset{(*)}{\Leftrightarrow} \frac{1}{\left(t + f''(0)^{-1/2} \right)^2} \leq f''(t) \leq \frac{1}{\left(-t + f''(0) \right)^2} \\
&\Leftrightarrow \frac{f''(0)}{\left(t f''(0)^{1/2} + 1 \right)^2} \leq f''(t) \leq \frac{f''(0)}{\left(-t f''(0)^{1/2} + 1 \right)^2},
\end{aligned}$$

where we assumed $t < f''(0)^{-1/2}$ in (*). This shows both claims.

Solution of the Exercise 4.7

We will start by proving an auxiliary result needed later: The newton decrement satisfies

$$\lambda(x) = \sup_{v \neq 0} \frac{-v^\top \nabla f(x)}{\left(v^\top \nabla^2 f(x) v \right)^{1/2}}. \tag{4.10}$$

We can write $\lambda(x)$ as

$$\begin{aligned}
\lambda(x) &= \left\| \nabla^2 f(x)^{-1/2} \nabla f(x) \right\|_2 \\
&= \sup_{\|u\|_2 = 1} -u^\top \nabla^2 f(x)^{-1/2} \nabla f(x)
\end{aligned}$$

where we used cauchy schwarz in the first inequality. By setting $v := \nabla^2 f^{-1/2} w$ we get

$$\lambda(x) = \sup_{v^\top \nabla^2 f(x) v = 1} \left(-v^\top \nabla f(x) \right)$$

and (4.10) follows by normalizing.

For the second step we first note that since f is strictly convex we did already reach f^* if $\nabla f(x) = 0$. Therefore we may assume $\nabla f(x) \neq 0$. Now let $v \in \mathbb{R}^n$ be a descent direction, i.e. $v^\top \nabla f(x) < 0$. Define $\tilde{f}(t) := f(x+tv)$ with t such that $(x+tv) \in U$. By definition $\tilde{f}$ is also self-concordant. Now we can integrate the lower bound from Proposition 4.7.8 for $\tilde{f}$ to get the lower bound

$$\tilde{f}'(t) \geq \tilde{f}'(0) + \tilde{f}''(0)^{1/2} - \frac{\tilde{f}''(0)^{1/2}}{1 + t\tilde{f}''(0)^{1/2}}.$$

Integrating another time yields the lower bound on $\tilde{f}(t)$

$$\tilde{f}(t) \geq \tilde{f}(0) + t\tilde{f}'(0) + t\tilde{f}''(0)^{1/2} - \log\left(1 + t\tilde{f}''(0)^{1/2}\right).$$

Differentiating the right hand side shows that it is minimized for

$$\bar{t} = \frac{-\tilde{f}'(0)}{\tilde{f}''(0) + \tilde{f}''(0)^{1/2}\tilde{f}'(0)}$$

and hence we can lower bound $\tilde{f}$ by

$$\begin{aligned}\inf_{t \geq 0} \tilde{f}(t) &\geq \tilde{f}(0) + \bar{t}\tilde{f}'(0) + \bar{t}\tilde{f}''(0)^{1/2} - \log\left(1 + \bar{t}\tilde{f}''(0)^{1/2}\right) \\ &= \tilde{f}(0) - \tilde{f}'(0)\tilde{f}''(0)^{-1/2} + \log\left(1 + \tilde{f}'(0)\tilde{f}''(0)^{-1/2}\right).\end{aligned}$$

Now we can use the auxiliary result (4.10) which shows

$$\lambda(x) \geq -\tilde{f}'(0)\tilde{f}''(0)^{-1/2}$$

and hence, if $\lambda(x) < 1$,

$$\inf_{t \geq 0} \tilde{f}(t) \geq \tilde{f}(0) + \lambda(x) + \log(1 - \lambda(x))$$

since $u + \log(1-u)$ is monotonically decreasing. Since this inequality holds for any descent direction, in particular

$$f^* \geq f(x) + \lambda(x) + \log(1 - \lambda(x)).$$

Now it can be shown that for all $\lambda \leq 0.68$

$$-(\lambda + \log(1-\lambda)) \leq \lambda^2$$

holds and hence we get

$$f^* \geq f(x) - \lambda(x)^2.$$

Solution of the Exercise 4.8

We start by defining $v \coloneqq -\nabla^2 f(x)^{-1}\nabla f(x)$ and hence the hint gives

$$(1-t\lambda(x))^2\nabla^2 f(x) \preceq \nabla^2 f(x+tv) \preceq \frac{1}{(1-t\lambda(x))^2}\nabla^2 f(x).$$

Now we can assume without loss of generality that $\nabla^2 f(x) = I_n$ and hence

$$(1-\lambda(x))^2 I_n \preceq \nabla^2 f(x^+) \preceq \frac{1}{(1-\lambda(x))^2} I_n.$$

This allows us to write $\lambda\left(x^+\right)$ as

$$\begin{aligned}
\lambda\left(x^+\right) &= \left\|\nabla^2 f\left(x^+\right)^{-1}\nabla f\left(x^+\right)\right\|_2 \\
&\leq (1-\lambda(x))^{-1}\left\|\nabla f\left(x^+\right)\right\|_2 \\
&= (1-\lambda(x))^{-1}\left\|\int_0^1 \nabla^2 f(x+tv)v\,dt + \nabla f(x)\right\|_2 \\
&= (1-\lambda(x))^{-1}\left\|\left(\int_0^1 \left(\nabla^2 f(x+tv) - I_n\right)\,dt\right)v\right\|_2 \\
&\leq (1-\lambda(x))^{-1}\left\|\left(\int_0^1 \left(\frac{1}{(1-t\lambda(x))^2} - 1\right)\,dt\right)v\right\|_2 \\
&\leq \|v\|_2 (1-\lambda(x))^{-1}\int_0^1 \left(\frac{1}{(1-t\lambda(x))^2} - 1\right)\,dt \\
&= \frac{\lambda(x)^2}{(1-\lambda(x))^2}.
\end{aligned}$$

This proves the exercise.

5. Constrained optimization

In this chapter we study algorithms for constrained optimization convex problems. In general, we are interested in programs of the form

$$\begin{aligned} &\min f(x) \\ &\quad g_i(x) \leq 0 \quad \text{for } i \in [m] \\ &\quad Ax = b \end{aligned} \tag{5.1}$$

where $f: \mathbb{R}^n \to \mathbb{R}$ and $g_i\ \mathbb{R}^n \to \mathbb{R}$ for $i \in [m]$ are twice differentiable function and $A \in \mathbb{R}^{\ell \times n}$ with **rank** $A = \ell < n$. Note that in the case of $m = \ell = 0$, that is in the case of unconstrained optimization convex problem, all the algorithms studied in the previous chapter can be successfully applied. Before we analyze the problem in full generality, we consider a case with equality constraints only.

5.1 Equality constrained minimization

In this section we discuss methods to solve the equality constrained minimization problem. More precisely, we solve the following problem

$$\begin{aligned} &\min f(x) \\ &\quad Ax = b \end{aligned} \tag{5.2}$$

where $f: \mathbb{R}^n \to \mathbb{R}$ is a twice differentiable function and $A \in \mathbb{R}^{\ell \times n}$ with **rank** $A = \ell < n$. The second assumptions means that there are fewer independent equality constraints than variables. There are several methods to approach equality constrained convex optimization problems.

The analytic method

Similar to the unconstrained case, there are few cases in which we can solve the problem analytically. Like the constrained case, the most important one is when f is a convex, quadratic function.

■ **Example 5.1.1** Let $P \in \mathscr{S}^n_+$ be a positive semidefinite matrix, and let $A \in \mathbb{R}^{\ell \times n}$. Consider the following problem

$$\begin{aligned} \min\ & f(x) = \frac{1}{2} x^\top P x + q^\top x + r \\ & Ax = b \end{aligned}$$

By Definition 3.4.1, the KKT optimality conditions for this problem are:

$$Ax^* = b, \qquad Px^* + q + A^\top v^* = 0.$$

We can rewrite those conditions as:

$$\begin{bmatrix} P & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ v^* \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix}$$

To solve the optimality conditions, we have to solve the set of $n+p$ linear equations in $n+p$ variables. Similarly, as in the unconstrained case, if a matrix P is nonsingular, there exists a unique solution to the system. When P is not positive definite (but still positive semidefinite, i.e., has the smallest eigenvalue equal to 0), any solution of the system is optimal. If there are no solutions to that system, the problem is unbounded from below. ■

In most of the cases, it is not possible to solve the equality constrained convex optimization problem analytically. In such cases, we have to use other methods. In the following, we briefly discuss three possible solutions.

The eliminating equality constraints method

A simple method to approach this problem is to eliminate the equality constraints from the system and then solve the unconstrained case using Newton's method. Similar approach for general convex problems was briefly discussed in Section 2.3.3. In this approach we first find a matrix $F \in \mathbb{R}^{n \times (n-\ell)}$ whose range is the null space of A and any solution $\hat{x} \in \mathbb{R}^n$ that satisfies $A\hat{x} = b$ that parametrize the affine feasible set

$$\{x |\ Ax = b\} = \{Fz + \hat{x} |\ z \in \mathbb{R}^{n-\ell}\}.$$

As a next step we solve an unconstrained optimization problem

$$\min f(Fz + \hat{x}) \tag{5.3}$$

Finally, given an optimal solution z^* to the unconstrained variant of the problem, we can compute an optimal solution to the original, constrained program as follows $x^* = Fz^* + \hat{x}$.

The dual method

Another method to construct the equality constrained convex optimization problem is to exploit duality theory. More precisely, assume that the Slater's condition holds so does the strong duality for the problem we consider. The dual function of the equality constrained minimization problem (5.2) is of the form

$$\begin{aligned} \widehat{L}(v) &= -b^\top v + \inf_x \left(f(x) + v^\top A x \right) \\ &= -b^\top v - \sup_x \left(-f(x) + \left(-A^\top v \right)^\top x \right) \\ &= -b^\top v - f^* \left(-A^\top v \right) \end{aligned}$$

where f^* is the conjugate of function f. Thus the dual problem is

$$\max \ -b^\top \nu - f^* \left(-A^\top \nu\right).$$

If the function $\widehat{L}$ is twice differentiable, then the methods for unconstrained optimization can be used to get an optimal dual solution. Moreover, since we assumed strong duality holds, an optimal dual solution has the same value as an optimal primal solution. We can use an optimal dual solution to compute an optimal primal solution. Indeed, let (λ^*, ν^*) be an optimal dual solution. The idea is to minimize over x the Lagrangian $L(x, \lambda^*, \nu^*)$ to get a point x^*. Then, if the solution x^* is primal feasible, it must also be primal optimal (by strong duality). If it is not primal feasible, then there is no primal feasible solution.

The feasible start Newton's method

Another approach to solving the equality constrained minimization problem is adjusting Newton's method for handling the equality constraints. This, in fact, is often a preferred method. In this chapter, we do not fully cover this method; however, we give a very short description to highlight adjustments to Newton's method that have to be performed in order to solve the equality constrained minimization problem (5.2).

An important initialization step in the feasible start Newton's method is to start with a feasible point, i.e., a point $x^{(0)}$ that satisfies $Ax^{(0)} = b$. Moreover, similar to Newton's method in the unconstrained minimization, we assume that the starting point is in the domain of the convex program. The feasible start Newton's method for equality constrained problem follows exactly the same steps as Newton's method for an unconstrained case, Algorithm 5, with the difference that some parameters are chosen differently. The main difference is that the descent direction $\Delta x_N^{(k)}$, step 3, is chosen differently. Indeed, recall that the idea behind Newton's method is to find an analytic solution of the second-order Taylor approximation of the function at a given point, i.e.

$$\min \tilde{f}(x^{(k)} + v) = f(x^{(k)}) + \nabla f(x^{(k)})^\top v + \frac{1}{2} v^\top \nabla^2 f(x^{(k)}) v$$

that, by the necessary optimality conditions (Proposition 9.2.9) gave a descent direction for the feasible start Newton's method of the form

$$\Delta x_N = -\frac{\nabla f(x)}{\nabla^2 f(x)}.$$

In the equality constrained case, this step has to be modified. The Newton step $\Delta_N^{(k)}$ at a feasible point $x^{(k)}$ is the solution of the following quadratic program

$$\begin{aligned} &\min \tilde{f}(x^{(k)} + v) = f(x^{(k)}) + \nabla f(x^{(k)})^\top v + \frac{1}{2} v^\top \nabla^2 f(x^{(k)}) v \\ &\ s.t. A(x^{(k)} + v) = b. \end{aligned}$$

that this is an optimization problem in variable $v \in \mathbb{R}^n$, which can be solved analytically as show in the Example 5.1.1. Note that the corresponding matrix in the KKT conditions must be non-singular. (It is not easy to see the fact why the matrix must be nonsingular, which is more restricted requirement than just asking the system to have a feasible solution as in Example 5.1.1. Details of this fact are omitted in this chapter as they come from the convergence analysis for the equality constrained minimization, which is also not presented here.) In a degenerated case of A having zero

rows, it corresponds exactly to the descent direction in Newton's method for an unconstrained case. Analogous to Example 5.1.1 the Newton step Δx_N is characterized by

$$\begin{bmatrix} \nabla^2 f(x) & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x_N \\ w \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ 0 \end{bmatrix},$$

where w is an associated optimal dual variable for the quadratic program.

Other parameters of the feasible start Newton's method are little affected. One just has to be careful with the formula used to compute them, i.e., Newton's decrement λ^2 can be computed in an analogous way as for the constrained case; however, one has to use Equation (4.5), that is:

$$\lambda(x)^2 = -\nabla f(x)^\top \Delta x_N$$

The quitting criterion and the computation of the step size, in step 8 of the feasible start Newton's method remains unchanged. In particular, Proposition 4.7.8 holds. Similarly to an unconstrained minimization, we note that if f is a convex quadratic function, Newton's method for the equality constrained minimization solves the problem exactly. Moreover, it does so analytically in only one iteration.

Interestingly, Newton's method applied for an unconstrained minimization problem that is obtained after applying the eliminating equality constraints method to the equality constrained minimization problem is equivalent to Newton's method for equality constrained minimization problem. We leave the proof of this remark as an Exercise.

To finalize this subsection, we would like to state, without proof, the result regarding the convergence speed of the feasible start Newton's method. Similarly, as for the unconstrained Newton's method, the case when the function f is self-concordant gives a much cleaner convergence speed that does not depend on constants characterizing the function f. In fact, in the case f is self-concordant, the convergence speed of the feasible start Newton's method is exactly the same as Newton's method for unconstrained minimization problems.

Theorem 5.1.2 Let f be s strongly convex self-concordant function. For every $\varepsilon > 0$, the feasible start Newton's method with a backtracking line search achieves $f(x^{(k)}) - f^* \leq \varepsilon$ for

$$k \geq \frac{f(x^{(0)}) - f^*}{\gamma} + \log\log\frac{1}{\varepsilon},$$

for γ being a constant from Theorem 4.7.10.

The infeasible start Newton's method

In this paragraph, we briefly mention a modification of a feasible start Newton's method for which the starting point $x^{(0)}$ does not have to satisfy the system of linear equalities $Ax = b$. However, it is still required, similar to the unconstrained case, that the starting point $x^{(0)}$ is in the domain of the convex program.

In contrast to the feasible start Newton's method for equality constrained minimization, in the infeasible start Newton's method there are some subtle differences in the construction of Newton's algorithm. In this section, we focus only on Newton's step and do not discuss the modifications of stopping criteria and line search step, as they mostly affect convergence analysis only, that is not a topic of discussion in this chapter anyway.

The main idea to compute Newton's step, given a point x that does not necessarily satisfy the system of linear equalities $Ax = b$ is to find a step Δx_N such that $x + \Delta x_N$ satisfy optimality conditions at least approximately. In particular, by KKT conditions, we want the point to satisfy approximately

$$A(x + \Delta x_N) = b \qquad \nabla f(x + \Delta x_N) + A^\top w = 0$$

where w is a dual variable associated to equality constrains. To make computation possible, we use the first order approximation of $\nabla f(x)\,(x+\Delta x_N) \approx \nabla f\,(x)+\nabla^2 f(x)\Delta x_N$ that leads to the following approximated optimality conditions

$$A\,(x+\Delta x_N) = b \qquad \nabla f\,(x)+\nabla^2 f(x)\Delta x_N + A^\top w = 0$$

which in the matrix form can be written as

$$\begin{bmatrix} \nabla^2 f(x) & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x_N \\ w \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ b - Ax \end{bmatrix}.$$

Note that in the case of point x is a feasible point, i.e., satisfies $Ax = b$, the above conditions coincide with the ones given for the feasible start Newton's method. In case x does not satisfy $Ax = b$, the second block component of the right-hand side is a slack vector measuring the infeasibility of the point x.

5.2 Inequality constrained minimization

In this section, we analyze algorithms to solve problem (5.1). We assume the functions satisfy the requirements listed at the beginning of this chapter. Moreover, we assume that the problem is strictly feasible, i.e., there exists a point $\bar{x} \in \mathbb{P}$ such that $A\bar{x} = b$ and $g_i(x) < 0$ for all $i \in [m]$. The existence of such a point ensures that Slater's condition holds and there is no duality gap. Moreover, since Slater's condition is satisfied, by Theorem 3.5.1 the dual optimal value is attained and by Proposition 3.4.2 any pair of primal optimal and dual optimal solutions $x^*, (\lambda^*, \nu^*)$ satisfy KKT conditions. Finally, we assume that the optimal solution is unique. We will use this assumption in the convergence analysis.

The logarithmic barrier function

In the following, we approach to construct an algorithm that will solve problem (5.1), which can be alternatively seen as a method to solve the KKT conditions. We start with an intuitive approach that goes back to the Duality chapter. Recall that the duality theory has been motivated by a technique to transform a constrained optimization problem into an unconstrained optimization using the indicator functions $I_-,\ I_0 :\ \mathbb{R} \to \mathbb{R} \cup \{\infty\}$ whose goal was to punish an infeasible solution with an infinity penalty term so that such a point is never considered as an optimal one. Since we know how to solve an unconstrained optimization problem with Newton's type methods that can be extended to equality constrained minimization problems, we apply this technique as a starting point. More precisely, we recall the definition of the indicator function I_- defined as:

$$I_-(x) = \begin{cases} 0 & \text{for } x \le 0, \\ \infty & \text{for } x > 0, \end{cases}$$

which allows to rewrite program (5.1) in the following form:

$$\begin{aligned} \min\ & f(x) + \sum_{i=1}^{m} I_-\left(g_i(x)\right) \\ & Ax = b. \end{aligned} \tag{5.4}$$

This way, we obtain an equality constrained minimization problem. However, since the objective function is convex but is not twice continuously differentiable, we cannot apply methods for equality constrained minimization problems.

Our main idea, similar as in the duality chapter, is to approximate the indicator function in a way that the function is twice continuously differentiable and that optimal solution of the program (5.4) is a good enough approximation of the optimal value of the problem (5.1).

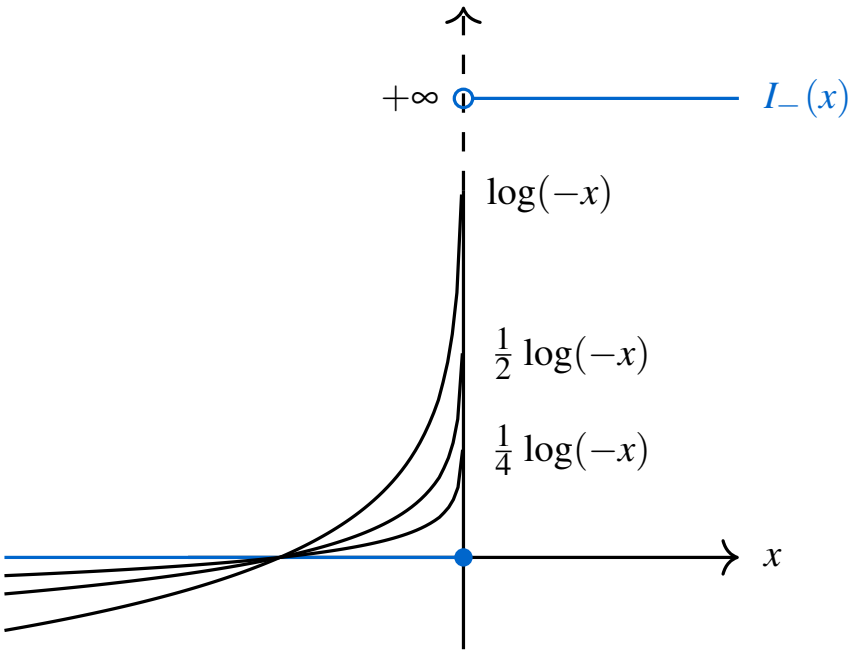


Figure 5.1: Logarithmic barrier functions for sample values of $t = \{1,2,4\}$ approximating an indicator function $I_-(x)$.

Definition 5.2.1 The function $\tilde{I}_- : \mathbb{R}_{<0} \to \mathbb{R}$, for $t > 0$, defined as

$$\tilde{I}_-(x) := -\frac{1}{t}\log(-x)$$

is called a *logarithmic barrier function* with parameter t.

The logarithmic barrier function, for sample values of parameter t can be seen in Figure 5.1.

Proposition 5.2.2 For every $t > 0$ the logarithmic barrier function is convex and twice continuously differentiable.

Proof. For every $t >$ the logarithmic barrier function is convex in the domain $(-\infty, 0)$ since the function $\log(x)$ is concave in the domain $(0, \infty)$. Moreover, the first derivative of the logarithmic barrier function is

$$f^{'}(x) = -\frac{1}{tx}$$

and the second derivative is

$$f^{''} = \frac{1}{tx^2}.$$

which is continuous function in the interval $[-\infty, 0)$. This implies that for every $t > 0$, the logarithmic barrier function is twice continuously differentiable in its domain. ■

The notation of logarithmic barrier functions can be extended to involve a set of constraints in the program 5.1 in the following way.

Definition 5.2.3 The function $\phi : \{x \in \mathbb{R}^n | \ g_i(x) \leq 0, i \in [m]\} \to \mathbb{R}$, for $t > 0$, defined as

$$\phi(x) := -\sum_{i=1}^{m} \log(-g(x_i))$$

is called a *logarithmic barrier function* for a program (5.1).

Using the logarithmic barrier function we can write an approximation of the program (5.1) parametrized by t in the following form

$$\begin{aligned} &\min \ f(x) - \frac{1}{t}\phi\left(-g_i(x)\right) \\ &s.t. \ Ax = b \end{aligned} \tag{5.5}$$

By Proposition 5.2.2 the above program is a convex program constrained with linear equalities. For any $t > 0$ the program can be solved using methods described in Section 5.1.

We finalize the section with several observations.

Observation 1: The logarithmic barrier function $\tilde{I}(x)$ approximates the indicator function $I_{-}(x)$ the better, the larger the value of parameter t is.

Observation 2: By Observation 1, quality of the approximation of program (5.5) increases with value of t.

Observation 3: The higher the value of the parameter t the more the Hessian varies for $g(x)$ close to zero, which might potentially affect the speed of the Newton's method.

Observation 4: Finally, we observe that the gradient and the Hessian of the logarithmic barrier function is given by

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

and

$$\nabla^2\phi(x) = \sum_{i=1}^{m} \frac{1}{g_i(x)^2} \nabla g_i(x) \nabla g_i(x)^\top + \sum_{i=1}^{m} \frac{1}{-g_i(x)} \nabla^2 g_i(x),$$

respectively.

Central path

In this section we investigate how the minimizers of the family of approximate optimization problems in (5.5) converge to the minimizer of the original problem (5.1). For the sake of simplicity in the analysis, in some cases we multiply the objective function by t that leads to the following optimization problem

$$\begin{aligned} &\min \ tf(x) + \phi\left(x\right) \\ &s.t. \ Ax = b \end{aligned} \tag{5.6}$$

Note, that multiplying the objective by t does not change the minimizers of the objective function.

Definition 5.2.4 A set of points $x_t^*, t > 0$ where x_t^* is a minimizer of the program (5.6), is called a *central path* of program (5.6).

■ **Example 5.2.5** Consider the following minimization problem

$$\begin{aligned} &\min \ -x \\ &s.t. \ x \leq 0 \end{aligned}$$

The above problem has no equality constraints, just for the simplicity of presentation, and only one inequality constraint. The objective function $f(x) = -x$. It is easy to see that the optimal value is 0

for $x = 0$. Our goal is to present how the family of programs 5.5 parametrized by t will converge to the optimal value. In particular, we want to observe how the central path in this particular case is constructed.

Let us first write down the approximated program (5.5) for our choice of functions $f(x) = -x$ and $g_1(x) = x$.

这里不清楚为什么g1(x) = x.

$$\min \quad -x - \frac{1}{t}\log(-x)$$

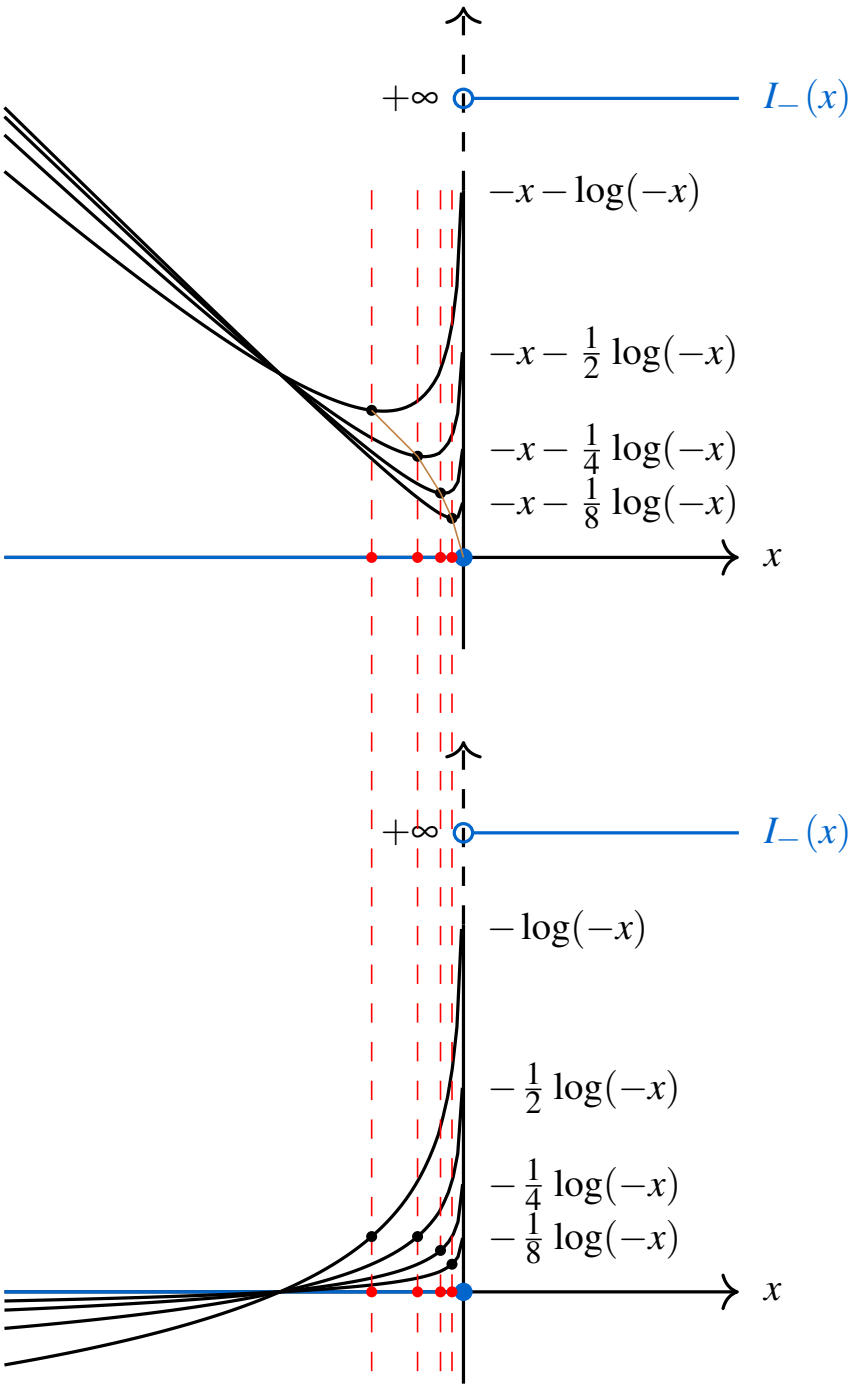
Here we do not multiply the objective function by t since multiplying by t does not change the minimizers of the objective function but changes its value, and for us, it is important to show that the central path converges to the optimal value of the original problem. Note that the objective function is convex and differentiable. By Proposition 9.2.9 the optimality conditions imply that the minimum is attained at x that satisfies

$$-\frac{1}{tx} - 1 = 0 \qquad \Rightarrow \qquad x = -\frac{1}{t}$$

and the optimal value is equal to

$$\frac{1}{t} - \frac{1}{t}\log(\frac{1}{t}) = \frac{1}{t}(1 + \log t).$$

The visualization of the central path for sample values of $t = \{1, 2, 4, 8\}$ can be seen on the figure below. In black, the objective function $f(x) = -x - \frac{1}{t}\log(-x)$ in the top, and the logarithmic barrier function $-\frac{1}{t}\log(-x)$ in the bottom. In blue, the indicator function $I_-(x)$. Red dots visualize the central path converging to the optimal solution $x = 0$. Black dots show the value of the approximated objective value $f(x) = \frac{1}{t}(1 + \log t)$ in the top and the logarithmic barrier function on the bottom. The brown lines shows how the values corresponding to the central path converge to the optimal solution of the original problem.



■

We can establish necessary and suffincet conditions for a point to be on the central path.

Proposition 5.2.6 For every $t > 0$, every point on the central path for the program (5.6) satisfies the following necessary conditions

$$Ax_t^* = b, \qquad g_i(x_t^*) < 0 \ \text{ for all } i \in [m].$$

Proof. Indeed, the first condition follows since every point on the central path is a minimizer of an approximate program parametrized by the same value of t and thus is a feasible point satisfying

equality constraints. The second condition follows from the fact that a point in the central path never satisfies inequality constraints with equality as this leads to an infinite punishment in the objective function, which contradicts the optimality of the points on the central path. ■

We can also establish sufficient conditions. For this, we start by introducing a concept of centrality conditions.

Definition 5.2.7 For a program (5.6) a point x_t satisfies a *central path conditions* if there exists a vector $\hat{\nu} \in \mathbb{R}^{\ell}$ that satisfies

$$t\nabla f(x_t) + \nabla \phi(x_t) + A^{\top}\hat{\nu} = 0.$$

Proposition 5.2.8 For every $t > 0$, a point x_t is in the central path for the program (5.6) if it satisfies the central path condition.

Proof. The proof follows by noting that a necessary condition and central path condition form a KKT conditions for the program (5.6) and applying Proposition 3.4.3. ■

As proof of Proposition 5.2.6 indicates, there is a close link between necessary conditions for a point to be in a central path, sufficient conditions given by central path conditions and a KKT point as defined in Definition 3.4.1. Indeed, for a program (5.1), one can think of the relationship that a point x belongs to the central path x_t^*, for $t > 0$ if and only if there exists λ, ν that satisfy

$$\begin{aligned} g_i(x) &\leq 0, && \text{for all } i \in [m], \\ Ax - b &= 0, && \\ \lambda_i &\geq 0, && \text{for all } i \in [m], \\ -\lambda_i g_i(x) &= \frac{1}{t}, && \text{for all } i \in [m], \\ \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x) + A^{\top}\nu &= 0, && \end{aligned} \tag{5.7}$$

which can be seen as satisfying KKT conditions approximately. The only difference between KKT conditions in Definition 3.4.1 is the KKT-4, that on the right-hand side takes value $1/t$ instead of 0.

The above reinterpretation of the central path conditions yields a construction of a dual feasible point for a program (5.1) for each point in the central path. The idea is summarized in the following Proposition.

Proposition 5.2.9 For each point x_t^* in a central path, a point (λ^*, ν^*) defined as

$$(\lambda_t^*)_i = -\frac{1}{t \cdot g_i(x_t^*)}, \qquad \nu_t^* = \frac{\nu}{t},$$

where ν is a point from Definition 5.2.7, is a dual feasible point for the problem (5.1).

Proof. First note that, by Proposition 5.2.6, $g_i(x_t^*) < 0$ for all $i \in [m]$, thus $(\lambda_t^*)_i > 0$. Moreover, we see that x_t^* minimizes the Lagrangian

$$L(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \nu^{\top}(Ax - b)$$

since for $\lambda = \lambda_t^*$ and $\nu = \nu^*$ the condition reduces to central path condition for a problem (5.1) before multiplying the objective by value t. ■

Proposition 5.2.9 implies that (λ^*, ν^*) is a dual feasible pair, for which the dual objective value is equal to:

$$\begin{aligned}\widehat{L}(\lambda^*, \nu^*) &= f(x_t^*) + \sum_{i=1}^{m} (\lambda_t^*)_i g_i(x_t^*) + (\nu^*)^\top (Ax^* - b) \\ &= f(x_t^*) - \frac{m}{t}.\end{aligned}$$

As a consequence we have a measure for suboptimality of the point x_t^* of the form

$$f(x_t^*) - f^* \leq \frac{m}{t}$$

which confirms the convergence with t going to $+\infty$.

The barrier method

Using Proposition 5.2.9 and the discussion afterwards, one can construct a very simple algorithm for an inequality constrained convex minimization problem. Note, that for every $t > 0$, a solution in the central path, is at most m/t far away from the optimal solution. This indicate a very natural and simple algorithm. Apply $t = m/\varepsilon$ and solve the equality constrained problem (5.5) to get a solution whose value is at most ε far away from the optimal value. This approach, thought very simple, is efficient only for reasonable values of ε, small m and good starting point. In all other cases it is very rarely used. However, from it is a cornerstone of other, more efficient methods. One of such methods, called the barrier method, we study in this section.

The barrier method, uses a very similar approach. However, instead of solving one problem for a fixed value of $t = m/\varepsilon$ it solves a sequence of problems for decreasingly small value of t, using an optimal solution form the previous step as a starting point for the next iteration. Formally the algorithm takes the following form

Algorithm 7: Barrier method

Input : Strictly feasible x^*, $t = t^{(0)}$, μ, tolerance $\varepsilon > 0$
output : point x s.t. $f(x) - f^* \leq \varepsilon$

$k \leftarrow 0$
repeat
  Centering step: starting at x^* compute $x_t^* \leftarrow \arg\min\{tf(x) + \phi(x) | \, Ax = b\}$
  $x^* \leftarrow x_t^*$
  if $m/\varepsilon \leq t$ **then**
    quit
  end
  $t \leftarrow \mu t$
  $k \leftarrow k + 1$
end;
return $x^{(k)}$

In the barrier method, at each iteration, for given t, we compute a central point x_t^* starting from a previously computed point and then increase t by a factor of μ. Computing a central point is called a centering step, or an inner iteration of the algorithm. It is usually performed by Newton's method. Repeating the loop until the stopping criterion is met is called an outer iteration of the algorithm. An initially proposed method in this section can be seen as just one outer iteration and an inner iteration computed for $t = m/\varepsilon$. We make the following observations about the barrier method:

Observation 1: At each centering step, there is no need of computing exact minimizers, they do not have other meaning than forming a converging sequence of points in the central path. However, note that inexact minimizers do not provide feasible dual variables.

Observation 2: Choice of μ forms a trade-off between many Newton's steps in one inner iterations and few outer iterations, for big values of μ, and the opposite for small values of μ.

Observation 3: Similar trade-off applies to the initial value of $t^{(0)}$ with many Newton's steps in the initial inner iteration for big values of $t^{(0)}$ and many outer iterations if the initial value $t^{(0)}$ is chosen too small.

It is easy to compute the number of outer iterations, that together with the convergence speed of Newton's method (e.g., from Theorem 4.6.2), gives an estimate of the convergence speed of the barrier method.

Proposition 5.2.10 For every $\varepsilon > 0$, the barrier method computes x^* such that $f(x^*) - f^* \leq \varepsilon$ after at most k steps, for every $k \in \mathbb{N}$ that satisfies

$$k \geq \frac{\log\left(\frac{m}{\varepsilon t}\right)}{\log \mu} + 1.$$

Proof. By Proposition 5.2.9 and the discussion afterwards, the barrier method computes a solution with ε accuracy once it reaches $t \geq \frac{m}{\varepsilon}$. The barrier method in Algorithm 7 first runs an initial centering step to reach the set the value $t := t^{(0)}$ and then in every step it increases the value of t by the multiplicative factor μ. After $k-1$ steps it reaches the value $t = m/\left(\mu^{k-1} t^{(0)}\right)$. Solving for k yields the bound

$$k \geq \frac{\log\left(\frac{m}{\varepsilon t}\right)}{\log \mu} + 1.$$

■

The above proposition gives a rough bound on the convergence speed. It is worth recalling that in order the bound to hold, assumptions on the function f listed in Theorem 4.6.2 must hold. Moreover, additional requirements, that have not been discussed in this book, to run equality constrained Newton's method must be met.

Feasibility and phase I method

One important note is, that barrier method needs a strictly feasible starting point. That is a nontrivial requirement. In this section we present several methods to deal with this problem. Finding a strictly feasible solution i usually called a phase I. Running the barrier method is called a phase II of the algorithm.

Basic method

Consider a problem (5.1). The basic method aims at finding a strictly feasible to problem (5.1) by solving a modified convex problem of the form:

$$\begin{aligned} &\min s \\ &\quad g_i(x) \leq s \quad \text{for } i \in [m] \\ &\quad Ax = b \end{aligned} \tag{5.8}$$

that is a problem in variables x and s. The purpose of adding a variable s is to measure infeasibility. To solve the problem, a barrier method can be applied with a strictly feasible starting point composed of any x in the domain of the problem (5.1) and $s > \max_{i\in[m]} g_i(x)$. Let s^* be the value of an optimal solution of the problem (5.8). The value of s^* decides on the existence of the strictly feasible solution to the problem (5.1) in the following sense:

- if $s^* <$ then the program (5.1) is strictly feasible and if (x,s), for $s < 0$ is feasible for program (5.8), then x is feasible for program (5.1).
- if $s > 0$ then program (5.1) is infeasible.
- if $s^* = 0$ then the program (5.1) does not have a strictly feasible point.

An important note has to be made regarding the practical implementation of the above cases. In the first case, when $s^* < 0$ there is no need in solving the program to optimality, it is enough to stop once the objective reaches a negative value. Similarly, in the second case, we can stop the computations once a dual feasible point is found with a positive objective. In the third case, the situation is much more complicated as in practice it might not be possible to determine if the objective value is exactly equal to zero. This can lead to problematic cases when deciding strict feasibility is not easily decidable. As a one famous example we refer the reader to a problem of deciding a feasibility of spectrahedron, that is not known to be polynomially solvable.

Phase I via infeasible start Newton method

Another approach to solve the issue with a lack of strictly feasible solutions as a starting point to the barrier method is to find such a point with infeasible start Newton's method. More precisely, we start with rewriting program (5.1) into an equivalent form involving one variable more

$$\begin{aligned} &\min f(x) \\ &\quad g_i(x) \leq s \quad \text{for } i \in [m] \\ &\quad Ax = b \\ &\quad s = 0 \end{aligned}$$

We use the infeasible start Newton's method to solve the problem

$$\begin{aligned} &\min t^{(0)} f(x) - \sum_{i=1}^{m} \log\left(s - g_i(x)\right) \\ &\quad Ax = b \\ &\quad s = 0 \end{aligned}$$

that can be initialized with any x in the domain of the program (5.1) and $s > \max_{i\in[m]} g_i(x)$.

5.3 Exercises

Exercise 5.1 Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be convex and twice differentiable, $A \in \mathbb{R}^{\ell\times n}, b \in \mathbb{R}^\ell$ with $\operatorname{rank}(A) = \ell < n$ and $Q \succeq 0$. Then the problem

$$\begin{aligned} &\min\ f(x) + (Ax-b)^\top Q\,(Ax-b) \\ &\text{s.t. } Ax = b \end{aligned} \tag{P_Q}$$

is equivalent to the original problem

$$\begin{aligned} &\min\ f(x) \\ &\text{s.t. } Ax = b. \end{aligned} \tag{P}$$

We assume that the optimal solution exists and is attained by x^*. Is the Newton step for (P_Q) the same as the Newton step for the original problem (P)?

Exercise 5.2 Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be convex and twice differentiable, $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{\ell\times n}, b \in \mathbb{R}^\ell$ with $\operatorname{rank}(A) = \ell < n$. The Newton decrement for the equality constrained problem is defined as

$$\lambda(x) = \left(\Delta x_{\text{nt}}^\top \nabla^2 f(x) \Delta x_{\text{nt}}\right)^{1/2}$$

where Δx_{nt} is defined by

$$\begin{pmatrix} \nabla^2 f(x) & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x_{\text{nt}} \\ w \end{pmatrix} = \begin{pmatrix} -\nabla f(x) \\ 0 \end{pmatrix}.$$

Prove

$$f(x) - \inf\left\{ \widehat{f}(x+v)\colon v \in \mathbb{R}^n,\ A(x+v) = b \right\} = \frac{\lambda(x)^2}{2},$$

where $\widehat{f}(x+v) = f(x) + \nabla f(x)^\top v + {}^1\!/_2 v^\top \nabla^2 f(x) v$.

Exercise 5.3 Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be convex and twice differentiable, $g_i\colon \mathbb{R}^n \to \mathbb{R}$ convex for $i \in [m]$ and $A \in \mathbb{R}^{\ell\times n}, b \in \mathbb{R}^\ell$. Suppose that the sublevel sets of

$$\begin{aligned} &\min\ f(x) \\ &\text{s.t. } g_i(x) \le 0, \qquad i \in [m] \\ &\quad Ax = b \end{aligned}$$

are bounded. Show that the sublevel sets of the associated centering problem

$$\begin{aligned} &\min\ t f(x) + \Phi(x) \\ &\text{s.t. } Ax = b \end{aligned}$$

are bounded. Here Φ denotes the log barrier $\Phi(x) = -\sum_{i=1}^m \log(-g_i(x))$.

Exercise 5.4 We consider a convex-concave game with inequality constraints,

$$\begin{aligned} &\min_w \max_z\ f(w,z) \\ &\qquad \text{s.t. } g_i(w) \le 0, \qquad i \in [m] \\ &\qquad\quad \tilde{g}_j(z) \le 0, \qquad j \in [\tilde{m}]. \end{aligned}$$

Here $w \in \mathbb{R}^n$ is the variable associated with minimizing the objective, and $z \in \mathbb{R}^{\tilde{n}}$ is the variable associated with maximizing the objective. The constraint functions g_i and $\tilde{g}_j$ are convex and differentiable, and the objective function $f\colon \mathbb{R}^n \times \mathbb{R}^{\tilde{n}}$ is differentiable and convex-concave, i.e. convex in w for each z, and concave in z for each w.
A *solution* or *saddle-point* for the game is a pair (w^*, z^*) for which

$$f(w^*, z) \leq f(w^*, z^*) \leq f(w, z^*),$$

for every feasible w, z.
In this exercise we show how to solve this game using an extension of the barrier method, and the infeasible start Newton method.

i) Let $t > 0$. Explain why the function

$$t f(w,z) - \sum_{i=1}^{m} \log\left(-g_i(w)\right) + \sum_{j=1}^{\tilde{m}} \log\left(-\tilde{g}_j(z)\right)$$

is convex-concave in (w,z). We will assume that it has a unique saddle-point, $(w^*(t), z^*(t))$, which can be found using the infeasible start Newton method.

ii) As in the barrier method for solving a convex optimization problem, we can derive a simple bound on the suboptimality of $(w^*(t), z^*(t))$, which depends only on the problem dimensions, and decreases to zero as t increases. Let W and Z denote the feasibility sets for w and z,

$$W = \left\{ w \in \mathbb{R}^n \mid g_i(w) \leq 0, i \in [m] \right\}, \qquad Z = \left\{ z \in \mathbb{R}^{\tilde{n}} \mid \tilde{g}_j(z) \leq 0, j \in [\tilde{m}] \right\}.$$

Show that

$$f\left(w^*(t), z^*(t)\right) \leq \inf_{w \in W} f\left(w, z^*(t)\right) + \frac{m}{t},$$
$$f\left(w^*(t), z^*(t)\right) \geq \sup_{z \in Z} f\left(w^*(t), z\right) - \frac{\tilde{m}}{t},$$

and therefore

$$\sup_{z \in Z} f\left(w^*(t), z\right) - \inf_{w \in W} f\left(w, z^*(t)\right) \leq \frac{m + \tilde{m}}{t}.$$

Exercise 5.5 — Programming Exercise. Implement the barrier method. Your code should take as an input the function to be minimized together with its gradient and Hessian - similar to the input on Gradient descent - together with functions and their gradients and Hessians for each of the constraints. You can assume that there are no equality constraints if you want to simplify the implementation.

Plot the feasibility region of the input program and all the intermediate points found to visualize the central path.

As a subroutine you need Newton's method. For this you can use the code on Gradient descent from the webpage as a basis. There is also a notebook with example programs as test input on the webpage.

5.4 Solutions

Solution to the Exercise 5.1

Fix $x \in \mathbb{R}^n$ and denote $g \coloneqq \nabla f(x), H \coloneqq \nabla^2 f(x)$. The Newton step for the new problem (P_Q) is given by

$$\begin{pmatrix} H + A^\top QA & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ w \end{pmatrix} = \begin{pmatrix} -g - 2A^\top QAx + 2A^\top Qb \\ 0 \end{pmatrix}.$$

Since the second equation shows $A\Delta x = 0$ we get

$$\begin{pmatrix} H & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ w \end{pmatrix} = \begin{pmatrix} -g - 2A^\top QAx + 2A^\top Qb \\ 0 \end{pmatrix}$$

and applying this another time finally yields

$$\begin{pmatrix} H & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} -g \\ 0 \end{pmatrix},$$

where $\tilde{w} = w + 2QAx - 2Qb$. This concludes the proof.

Solution to the Exercise 5.2

By the definition of Δx_{nt} we get $A\Delta x_{\text{nt}} = 0$ and hence

$$\Delta x_{\text{nt}}^\top \nabla^2 f(x) \Delta x_{\text{nt}} = -\nabla f(x)^\top \Delta x_{\text{nt}}.$$

Therefore we get

$$\begin{aligned} \widehat{f}(x + \Delta x_{\text{nt}}) &= f(x) + \nabla f(x)^\top \Delta x_{\text{nt}} + \frac{1}{2} \Delta x_{\text{nt}}^\top \nabla^2 f(x) \Delta x_{\text{nt}} \\ &= f(x) + \frac{1}{2} \nabla f(x)^\top \Delta x_{\text{nt}} \\ &= f(x) - \frac{1}{2} \lambda(x)^2. \end{aligned}$$

Since Δx_{nt} is the minimizer, this yields the result.

Solution to the Exercise 5.3

Suppose there exists an unbounded sublevel set $S = \{ x \in X \mid tf(x) + \Phi(x) \leq M \}$, where X is the feasible set. By convexity of f and X we get that there exists a ray

$$R = \{ x + sv \colon s \geq 0 \}$$

with $v \neq 0$ and x strictly feasible that is contained in S. It follows that $Ax = b, Av = 0$ and $g_i(x + sv) < 0$ for all $i \in [m]$. By assumption the sublevel sets of f are bounded and hence $f(x + sv)$ must increase in s for sufficiently large s. Now we can without loss of generality assume that $\nabla f(x)^\top v > 0$ and calculate

$$\begin{aligned} M &\geq tf(x + sv) - \sum_{i=1}^m \log\left(-g_i(x + sv)\right) \\ &\geq tf(x) + st\nabla f(x)^\top v - \sum_{i=1}^m \log\left(-g_i(x) - s\nabla g_i(x)^\top v\right) \end{aligned}$$

for all $s \geq 0$. This is however impossible due to $\nabla f(x)^\top v > 0$.

Solution to the Exercise 5.4

i) Follows from the convex-concave property of f; convexity of $-\log(-g_i)$, and concavity of $\log(-\tilde{g}_j)$.

ii) Since $(w^*(t), z^*(t))$ is a saddle-point of

$$tf(w,z) - \sum_{i=1}^{m} \log(-g_i(w)) + \sum_{j=1}^{\tilde{m}} \log(-\tilde{g}_j(z)),$$

its gradient with respect to w, and also with respect z, vanishes there:

$$t\nabla_w f(w^*(t), z^*(t)) + \sum_{i=1}^{m} \frac{1}{-g_i(w^*(t))} \nabla g_i(w^*(t)) = 0,$$
$$t\nabla_z f(w^*(t), z^*(t)) + \sum_{j=1}^{\tilde{m}} \frac{-1}{-\tilde{g}_j(z^*(t))} \nabla \tilde{g}_j(z^*(t)) = 0.$$

It follows that $w^*(t)$ minimizes

$$f(w, z^*(t)) + \sum_{i=1}^{m} \lambda_i g_i(w)$$

over w, where $\lambda_i = 1/(-t g_i(w^*(t)))$, i.e. for all $w \in W$ we have

$$f(w^*(t), z^*(t)) + \sum_{i=1}^{m} \lambda_i g_i(w^*(t)) \leq f(w, z^*(t)) + \sum_{i=1}^{m} \lambda_i g_i(w).$$

The left hand side is equal to $f(w^*(t), z^*(t)) - m/t$, and for all $w \in W$, the second term on the righthand side is nonpositive, so we have

$$f(w^*(t), z^*(t)) \leq \inf_{w \in W} f(w, z^*(t)) + \frac{m}{t}.$$

A similar argument shows

$$f(w^*(t), z^*(t)) \geq \sup_{z \in Z} f(w^*(t), z) - \frac{m}{t}.$$

III Part Three

6. Second-order cone programs

In this chapter, we study the first important type of conic programs, namely, second-order cone programs (SOCP). We start with defining the Lorentz cone (second-order cone) and prove it is both self-dual and proper, which leads to the definition of the second-order cone constraints that are a building block of SOCPs. Next, we discuss several special cases of SOCPs, including quadratic programs and quadratically constrained quadratic programs. Finally, we present several applications of second-order cone programming.

6.1 Second-order cone

We start with the following definition.

Definition 6.1.1 — Light cone. Let $S \subseteq \mathbb{R}^n$ be a convex set. The set

$$\{(x,c) \in \mathbb{R}^n \times \mathbb{R}_{\geq 0} : \ x \in cS\}$$

is a convex cone called the *light cone*, generated by S.

There are many possible constructions of light cones. However, the most natural one is when the set S is an n-dimensional unit ball. This cone is called the second-order cone, also known as the Lorentz cone.

Definition 6.1.2 — Second-order cone, Lorentz cone. The set

$$\mathcal{Q}^n := \{(x,c) \in \mathbb{R}^n \times \mathbb{R}_{\geq 0} : \ \|x\|_2 \leq c\}$$

is called the *second-order cone* or the *Lorentz cone*.

A very useful variant of the second-order cone is the rotated second-order cone. It is formally defined in the following way:

Definition 6.1.3 — Rotated second-order cone. The set

$$\mathcal{Q}^n_{rot} := \left\{(x,y,z) \in \mathbb{R}^n \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} : \; 2yz \geq \|x\|_2^2\right\}$$

is called the *rotated second-order cone*.

The rotated second-order cone can be converted into an ordinary second-order cone by the following rule.

Theorem 6.1.4 For every $n \in \mathbb{N}$, $n \geq 1$ the point $(x,y,z) \in \mathbb{R}^n \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ belongs to $\mathcal{Q}^n_{rot}$ if and only if the point $\left(\begin{pmatrix} y-z \\ \sqrt{2}x \end{pmatrix}, y+z\right) \in \mathbb{R}^{n+1} \times \mathbb{R}_{\geq 0}$ belongs to $\mathcal{Q}^{n+1}$.

Proof. Let $\overline{\mathcal{Q}}^n_{rot} \subseteq \mathbb{R}^{n+1}$ be the set defined as follows:

$$\overline{\mathcal{Q}}^n_{rot} := \left\{(x,y,z) \in \mathbb{R}^n \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} : \; y+z \geq \left\|\begin{matrix} y-z \\ \sqrt{2}x \end{matrix}\right\|_2\right\}.$$

First we show that the above set is exactly the rotated second-order cone, that is for every $n \in \mathbb{N}$, $n \geq 1$ we have $\mathcal{Q}^n_{rot} = \overline{\mathcal{Q}}^n_{rot}$.

Indeed, let $(x,y,z) \in \mathcal{Q}^n_{rot}$, then $2yz \geq \|x\|_2^2$. Multiplying the constrain by 2, we get that $4yz \geq 2\|x\|_2^2$, which can be rewritten as:

$$y^2 + 2yz + z^2 \geq y^2 - 2yz + z^2 + 2\|x\|_2^2,$$

which is equivalent to

$$(y+z)^2 \geq \left\|\begin{matrix} y-z \\ \sqrt{2}x \end{matrix}\right\|_2^2.$$

Since both sides of the latter inequality are nonnegative, this is equivalent to the condition

$$y+z \geq \left\|\begin{matrix} y-z \\ \sqrt{2}x \end{matrix}\right\|_2$$

thus $\mathcal{Q}^n_{rot} \subseteq \overline{\mathcal{Q}}^n_{rot}$. The steps can be reversed to prove $\overline{\mathcal{Q}}^n_{rot} \subseteq \mathcal{Q}^n_{rot}$.

To finish the proof, we have to show that for every $n \in \mathbb{N}$, $n \geq 1$ the point $(x,y,z) \in \mathbb{R}^n \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ belongs to $\overline{\mathcal{Q}}^n_{rot}$ if and only if the point $\left(\begin{pmatrix} y-z \\ \sqrt{2}x \end{pmatrix}, y+z\right) \in \mathbb{R}^{n+1} \times \mathbb{R}_{\geq 0}$ belongs to $\mathcal{Q}^{n+1}$.

We start with proving the only if direction. We have to show that for every $n \in \mathbb{N}$, $n \geq 1$, if the point (x,y,z) belongs to $\overline{\mathcal{Q}}^n_{rot}$ then the point $\left(\begin{pmatrix} y-z \\ \sqrt{2}x \end{pmatrix}, y+z\right)$ belongs to $\mathcal{Q}^{n+1}$. We have $y+z \in \mathbb{R}_{\geq 0}$, since $y,z \geq 0$ and

$$\left\|\begin{matrix} y-z \\ \sqrt{2}x \end{matrix}\right\|_2 \leq y+z$$

by the definition of the cone $\overline{\mathcal{Q}}^n_{rot}$.

Next we prove that if the point $\left(\begin{pmatrix} y-z \\ \sqrt{2}x \end{pmatrix}, y+z\right)$ belongs to $\mathcal{Q}^{n+1}$, then the point (x,y,z) belongs to $\overline{\mathcal{Q}}^n_{rot}$. Clearly, by the definition of the cone $\mathcal{Q}^{n+1}$ we have

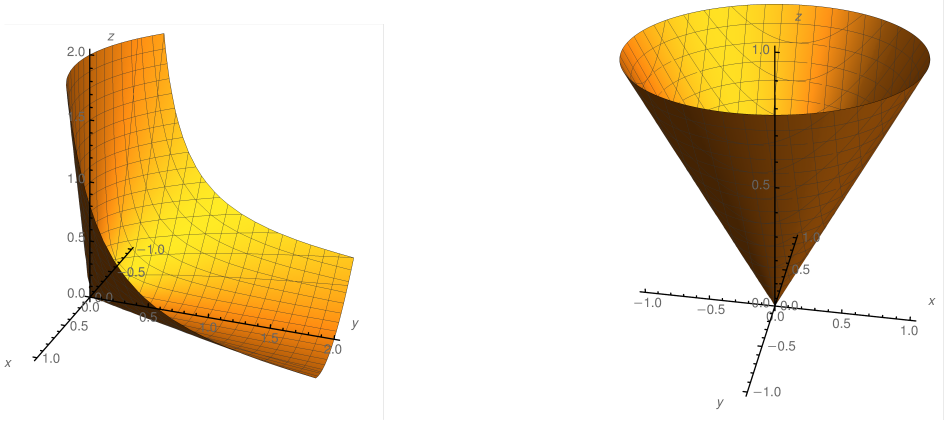
$$\left\|\begin{matrix} y-z \\ \sqrt{2}x \end{matrix}\right\|_2 \leq y+z$$

so it remains to show that y and z are nonnegative. We have

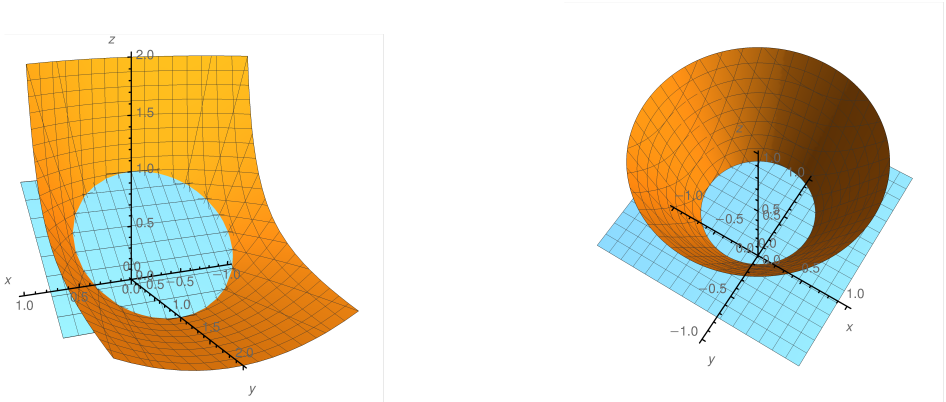
$$y+z \geq \left\|\begin{matrix} y-z \\ \sqrt{2}x \end{matrix}\right\|_2 \geq \sqrt{(y-z)^2} = |y-z|$$

which implies $y+z \geq y-z$ and $y+z \geq z-y$, which implies $y \geq 0$ and $z \geq 0$. ■

■ **Example 6.1.5** The rotated second-order cone in $\mathcal{Q}^1_{rot}$ on the left and the second-order cone in $\mathcal{Q}^2$ on the right, as subsets of $\mathbb{R}^3$.



The rotated second-order cone in $\mathcal{Q}^1_{rot}$ intersected with the affine hyperplane $y+z=1$ on the left and the second-order cone in $\mathcal{Q}^2$ intersected with the affine hyperplane $z=1$ on the right.



■

The second-order cone has many important properties, i.e., the cone $\mathcal{Q}^n$ is proper and self-dual. The first fact is left as an Exercise 6.1. The second fact is proved in the following theorem.

Theorem 6.1.6 For every $n \in \mathbb{N}_{\geq 1}$ the second-order cone $\mathcal{Q}^n$ is self-dual.

Proof. We want to prove that $(\mathcal{Q}^n)^* = \mathcal{Q}^n$. Recall that for a cone $K \subseteq \mathbb{R}^n$ its dual cone is defined as $K^* := \{l \in \mathbb{R}^n : \langle l, x\rangle \geq 0, \text{ for all } x \in K\}$.

We start with proving that $(\mathcal{Q}^n)^* \subseteq \mathcal{Q}^n$. Consider a point in $(x^*, c^*) \in \mathbb{R}^n \times \mathbb{R}$, such that $\langle (x^*, c^*), (x, c)\rangle = (x^*)^\top x + (c^*)^\top c \geq 0$, for every $(x, c) \in \mathcal{Q}^n$. We need to prove that $(x^*, c^*) \in \mathcal{Q}^n$.

First we prove that c^* is nonnegative, indeed by choosing $x = (0, \ldots, 0)$ and $c = 1$, we have $(x^*, c^*)^\top (x, c) = c^* \geq 0$.

Next we prove that $c^* \geq \|x^*\|_2$.

- If $x^* = (0, \ldots, 0)$, take $x = (0, \ldots, 0)$, $c = 1$ and by the above argument we have $c^* \geq 0 = \|x^*\|_2$.
- Otherwise, choose $x = -x^*$ and $c = \|x^*\|_2$ (note that $c \geq \|x\|_2$). Then $0 \leq (x^*, c^*)^\top (x, c) = -\|x^*\|_2 \|x^*\|_2 + c^* \|x^*\|_2$, which implies that $c^* \geq \|x^*\|_2$.

Next we prove that $(\mathcal{Q}^n)^* \supseteq \mathcal{Q}^n$. Consider a point $(x,c) \in \mathcal{Q}^n$. It is enough to prove that for every point $(x^*,c^*) \in \mathcal{Q}^n$ we have $(x^*,c^*)^\top (x,c) \geq 0$, which implies that $(x,c) \in (\mathcal{Q}^n)^*$. Indeed, By Cauchy-Schwarz inequality $(|\langle u,v\rangle|^2 \leq \langle u,u\rangle \langle v,v\rangle)$ we have $\langle x^*,x\rangle \geq -\|x^*\|_2 \|x\|_2$ and thus

$$(x^*,c^*)^\top (x,c) = c^*c + \langle x^*,x\rangle \geq c^*c - \|x^*\|_2 \|x\|_2$$

which is nonnegative, since we have $c^* \geq \|x^*\|_2$ and $c \geq \|x\|_2$.

■

6.2 Second-order cone programs

We start by introducing a special type of constraint, called second-order cone constraint. Second-order cone constraints are the building blocks of second-order cone programs.

Definition 6.2.1 — Second-order cone constraint. The *second-order cone constraint* is an inequality of the form

$$\|Ax+b\|_2 \leq c^\top x + d.$$

for some $A \in \mathbb{R}^{n_0 \times n}$, $b \in \mathbb{R}^{n_0}$, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$.

Satisfying the second-order cone constraints is equivalent to requiring the affine function $(Ax+b, c^\top x+d)$ to lie in the second-order cone $\mathcal{Q}^{n_0}$. Note that second-order cone constraint generalizes linear constraints, by setting $A = 0$ and $b = 0$. Now we are ready to present the second-order cone programs.

Definition 6.2.2 — Second-order cone program. Let $f \in \mathbb{R}^n$, $A_i \in \mathbb{R}^{n_i \times n}$, $b_i \in \mathbb{R}^{n_i}$, $c_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, for $i \in [m]$ and $F \in \mathbb{R}^{p \times n}$, $g \in \mathbb{R}^p$. The *Second-Order Cone Program (SOCP)* in a general form takes the form:

$$\begin{aligned} \min\ & f^\top x \\ & \|A_i x + b_i\|_2 \leq c_i^\top x + d_i \qquad \text{for } i = \{1,\dots m\} \\ & Fx = g. \end{aligned}$$

General form of the second-order cone program as defined in Definition 6.2.2 can be rewritten to standard form in the following way:

$$\begin{aligned} \min\ & f^\top x \\ & \left(A_i x + b_i, c_i^\top x + d_i\right) \in \mathcal{Q}^{n_i} \qquad \text{for } i = \{1,\dots m\} \\ & Fx = g. \end{aligned}$$

Several optimization programs can be formulated as second-order cone programs in a particular form. We start by showing that linear programs are special cases of second-order cone programs.

Theorem 6.2.3 Every linear program is a second-order cone program.

Proof. Consider a linear program as in Definition 1.4.1. Note that every linear program in the standard and the canonical form can be transformed into a linear program in general form (see

Exercise 1.11). Given an LP in general form:

$$\begin{aligned}\min\ & \overline{c}^\top x\\ & \overline{A}x \geq \overline{b}\\ & \overline{C}x \leq \overline{d}\\ & \overline{E}x = \overline{f}\end{aligned}$$

where $\overline{A},\overline{C},\overline{E} \in \mathbb{R}^{m\times n}$, $\overline{c} \in \mathbb{R}^n$ and $\overline{b},\overline{d},\overline{f} \in \mathbb{R}^m$ we can multiply the constraint $\overline{A}x \leq \overline{b}$ by -1 and then concatenate the first and the second constraint (note that this step is analogous to the proof that every linear program in general form can be transform to the linear program in standard form). Let $C := (-\overline{A}^\top | \overline{C}^\top)^\top$, $d := -(-\overline{b}^\top | \overline{d}^\top)^\top$ and $c = \overline{c}$, $f = \overline{f}$, $E = \overline{E}$ then get the following equivalent linear program.

$$\begin{aligned}\min\ & c^\top x\\ & 0 \geq Cx + d\\ & Ex = f\end{aligned}$$

which is the second-order cone program as defined in Definition 6.2.2. ■

6.3 Duality for second-order cone programs

In this section we will derive the dual of second-order cone programs, starting with the general form. Therefore we first introduce new variables y_i, t_i and rewrite the the general form to

$$\begin{aligned}\min\ & f^\top x\\ & \|y_i\|_2 \leq t_i \qquad \text{for } i = \{1,\dots m\}\\ & y_i = A_i x + b_i \qquad \text{for } i = \{1,\dots m\}\\ & t_i = c_i^\top x + d_i \qquad \text{for } i = \{1,\dots m\}\\ & Fx = g.\end{aligned}$$

This gives us the Lagrangian

$$\begin{aligned}&L(x,y,t,\lambda,\nu,\tilde{\nu},\bar{\nu})\\ &= f^\top x + \sum_{i=1}^m \lambda_i (\|y_i\|_2 - t_i) + \sum_{i=1}^m \tilde{\nu}_i^\top (y_i - A_i x - b_i) + \sum_{i=1}^m \bar{\nu}_i \left(t_i - c_i^\top x - d_i\right) + \nu^\top (Fx - g)\\ &= \underbrace{\left(f - \sum_{i=1}^m A_i \tilde{\nu}_i^\top - \sum_{i=1}^m \bar{\nu}_i c_i + F^\top \nu\right)^\top}_{(1)} x + \underbrace{\sum_{i=1}^m \left(\lambda_i \|y_i\|_2 + \tilde{\nu}_i^\top y_i\right)}_{(2)} + \underbrace{\sum_{i=1}^m (\bar{\nu}_i t_i - \lambda_i t_i)}_{(3)} + C,\end{aligned}$$

where C is a constant not depending on x,y,t. To calculate the Lagrange dual function we now minimize the seperated terms one by one. First it is clear that (1) is unbounded from below for any values but

$$f = \sum_{i=1}^m A_i \tilde{\nu}_i^\top + \sum_{i=1}^m \bar{\nu}_i c_i - F^\top \nu.$$

For the second term we first note that it is lower bounded by 0 for $\|\tilde{\nu}_i\| \leq \lambda_i$. If $\|\tilde{\nu}_i\| > \lambda_i$ on the other hand, choosing $y_i = -s\tilde{\nu}_i$ for some $s > 0$ shows that (2) is unbounded from below by letting s

tend to infinity. Finally (3) is bounded from below if and only if $\lambda_i = \bar{\nu}_i$. Therefore the Lagrange dual function is given by

$$\widehat{L}(\lambda, \tilde{\nu}, \bar{\nu}, \nu) = \begin{cases} C = -\sum_{i=1}^{m} \left(\tilde{\nu}_i^\top b_i + \lambda_i d_i + \nu_i g_i \right), & \text{if } f = \sum_{i=1}^{m} A_i \tilde{\nu}_i^\top + \sum_{i=1}^{m} \bar{\nu}_i c_i - F^\top \nu, \\ & \|\tilde{\nu}_i\|_2 \leq \lambda_i \text{ and } \lambda = \bar{\nu} \\ -\infty, & \text{otherwise.} \end{cases}$$

Summing up we derived the dual problem of a SOCP in general form to be

$$\begin{aligned} \max_{\lambda, \tilde{\nu}, \nu} \; & -\sum_{i=1}^{m} \left(\tilde{\nu}_i^\top b_i + \lambda_i d_i + \nu_i g_i \right) \\ \text{s.t.} \; & \sum_{i=1}^{m} A_i \tilde{\nu}_i^\top + \sum_{i=1}^{m} \bar{\nu}_i c_i - F^\top \nu = f \\ & \|\tilde{\nu}_i\|_2 \leq \lambda_i, \end{aligned}$$

which again is a SOCP in general form.

For SOCPs in standard form on the other hand we get the following result by the conic duality in standard form, Definition 2.2.5, and because $\mathcal{Q}_n$ is self-dual.

Definition 6.3.1 — Second-Order Cone Program duality. Let $A \in \mathbb{R}^{m \times (n+1)}$, $c, b \in R^{n+1}$. The primal and the *dual* formulation of the second-order cone program (SOCP) in standard form are the following:

$$\begin{aligned} &\textit{Primal}: && \textit{Dual}: \\ &\min c^\top x && \max b^\top y \\ &\quad Ax = b, && \quad c - A^\top y \in \mathcal{Q}^n \\ &\quad x \in \mathcal{Q}^n \end{aligned}$$

R As for the general conic programs, weak duality holds for the second-order cone programs. Let x, y be feasible solutions to the primal and the dual formulation of the second-order cone program in standard form, respectively. Then

$$c^\top x \geq b^\top y.$$

6.4 Quadratic optimization problems

Linear programs are not the only class of optimization problems that can be solved using second-order cone programs. Optimizing a quadratic function over a polyhedron can be done by transforming the problem into the second-order cone program. Moreover, second-order cone programs capture the problem of optimizing a quadratic function over the feasibility set described with inequalities involving convex quadratic function. We study this topic in the following section.

Definition 6.4.1 Let $Q \in \mathcal{S}_+^n$ be a symmetric positive semidefinite matrix and let $c \in \mathbb{R}^n$, $C \in \mathbb{R}^{m \times n}$ and $d \in \mathbb{R}^m$, $A \in \mathbb{R}^{\ell \times n}$ and $b \in \mathbb{R}^\ell$. The *quadratic program (QP)* in general form has

the form:

$$\begin{aligned} \min\ & x^\top Qx + c^\top x \\ & Cx \le d \\ & Ax = b. \end{aligned}$$

Note that in Definition 6.4.1 the matrix Q is positive semidefinite, thus the objective function is convex. We can view QP as a special case of SOCP. To do so first rewrite QP in the following form.

$$\begin{aligned} \min\ & q + c^\top x \\ & x^\top Qx \le q \\ & Cx \le d \\ & Ax = b. \end{aligned}$$

and note that $x^\top Qx \le q$ if and only if $(Q^{1/2}x, q, 1/2) \in \mathscr{Q}^n_{rot}$. Indeed, by Fact 1 in the proof of Theorem 7.1.3 for the positive semidefinite matrix Q there exists a square root $Q^{1/2}$ such that $Q = Q^{1/2}Q^{1/2}$. That is why we can write $x^\top Qx = \left\|Q^{1/2}x\right\|_2^2$ thus $x^\top Qx \le q$ if and only if $\left\|Q^{1/2}x\right\|_2^2 \le q$.

6.5 Quadratically constrained quadratic optimization problems

Definition 6.5.1 Let $Q, Q_1, \ldots Q_{m_1} \in \mathscr{S}^n_+$ be symmetric positive semidefinite matrices and let $c \in \mathbb{R}^n$, $p_i \in \mathbb{R}^n, r_i \in \mathbb{R}$ for $i \in [m_1]$, $C \in \mathbb{R}^{m_2 \times n}$ and $d \in \mathbb{R}^{m_2}$. Further let $A \in \mathbb{R}^{\ell \times n}$ and $b \in \mathbb{R}^\ell$. The *quadratically constrained quadratic program* in general form is the following program:

$$\begin{aligned} \min\ & x^\top Qx + c^\top x \\ & x^\top Q_i x + 2p_i^\top x + r_i \le 0 \qquad \text{for } i \in [m_1] \\ & Cx \le d \\ & Ax = b. \end{aligned}$$

6.6 Applications

In this section, we show a few examples of the quadratically constrained quadratic program. The first one is portfolio optimization.

Markowitz portfolio optimization

■ **Example 6.6.1** Consider a problem of managing a portfolio of n assets over a given period of time (e.g., for one month). Given a budget B the goal is to decide which assets we should buy for a given period of time.

First, consider the case where a relative price change of assets is known for the given period of time. For asset $i \in [n]$ denote the change in price divided by its price at the beginning of the period by p_i, e.g., $p_i = 2$ means that the i'th asset has doubled its value over a given period of time. In this case, the goal is to choose the subset of assets to maximize the return at the end of the period. This can be done with a simple linear program of the following form:

$$\begin{aligned} \max\ & p^\top x \\ & \frac{1}{B}\mathbf{1}^\top x = 1 \\ & x \ge 0 \qquad \text{for } i \in [n]. \end{aligned}$$

The variable x_i describes the fraction of the budget B devoted to buying asset i. The above problem has a simple solution: spending the whole budget B on the asset that gives the most significant return. The linear program technique could be helpful in some more advanced cases where additional linear constraints are required to model the problem. However, the case we are interested in requires more advanced tools to solve the problem.

Consider a setting where the change in price vector of assets $p \in \mathbb{R}^n$ is a random variable. The mean value of the return is $\overline{p}^\top x$ and the variance is $x^\top \Sigma x$. The choice of the assets to buy requires a balance between maximizing the expected return and minimizing the risk, that is, the variance. In the classical Markowitz portfolio optimization problem, the objective is to minimize the risk for a given fixed return value r at the end of the period. This can be modeled with the following program:

$$\begin{aligned} \min\ & x^\top \Sigma x \\ & \overline{p}^\top x \geq r \\ & \frac{1}{B}\mathbf{1}^\top x = 1 \\ & x \geq 0 \qquad \text{for } i \in [n] \end{aligned}$$

which is a quadratic program.

An interesting question, from the practical perspective, is how to compute the covariance matrix. Given observations

$$\begin{aligned} & X_{11}, X_{12}, \ldots, X_{1N} && \text{of asset 1} \\ & X_{21}, X_{22}, \ldots, X_{2N} && \text{of asset 1} \\ & \qquad\qquad \vdots \\ & X_{n1}, X_{n2}, \ldots, X_{nN} && \text{of asset n} \end{aligned}$$

the (i, j) entry of the convariance matrix at the current moment in time is computed by:

$$\Sigma_{i,j} = \frac{\sum_{k=1}^{N} \left(X_{ik} - \bar{X}_i\right)\left(X_{jk} - \bar{X}_j\right)}{N-1}$$

for $\bar{X}_i = \frac{1}{N}\sum_{k=1}^{N} X_{ik}$. In practice, in many cases, the covariance matrix at the current moment is used as a predictor for the covariance matrix for the future. The reason behind this is that the volatility of an asset is believed to be unchanged over time. Predicting an expected return is a different story. Computing a return of an asset at a current moment, based on historical data, is easy. However, it is not a good predictor for the future expected return. This is one of the factors that make portfolio management very challenging. ∎

Markowitz portfolio optimization with loss risk constraints

Consider an extension of the classical Markowitz portfolio optimization. Assume that the change in price vector $p \in \mathbb{R}^n$ is a Gaussian random variable. This implies that the return is Gaussian variable with mean $\overline{p}^\top x$ and variance $x^\top \Sigma x$. Consider a *loss risk constraint* of the form

$$P(r < \alpha) \leq \beta$$

that describes the maximum probability β of an unacceptable return α, that is equivalent to

$$\overline{p}^\top x + \phi^{-1}(\beta)\left\|\Sigma^{1/2} x\right\|_2 \geq \alpha$$

where ϕ is the cumulative distribution function of a unit Gaussian random variable. For $\beta \leq \frac{1}{2}$, and thus $\phi^{-1}(\beta) \leq 0$ (note that for $\beta > \frac{1}{2}$ the constraint becomes nonconvex in x), this can be formulated as the second-order cone program of the following form

$$\begin{aligned}
\max\ & \overline{p}^\top x \\
& \overline{p}^\top x + \phi^{-1}(\beta) \left\| \Sigma^{1/2} x \right\|_2 \geq \alpha, \\
& \frac{1}{B} \mathbf{1}^\top x = 1, \\
& x_i \geq 0 \qquad \text{for } i \in [n].
\end{aligned}$$

Distance between polyhedra

Let $P_1 = \{x \in \mathbb{R}^n : \ A_1 x \leq b_1\}$ and $P_2 = \{x \in \mathbb{R}^n : \ A_2 x \leq b_2\}$ be polyhedra for some matrices $A_1, A_2 \in \mathbb{R}^{m \times n}$ and $b_1, b_2 \in \mathbb{R}^m$. The distance between P_1 and P_2 defined as

$$\text{dist}(P_1, P_2) = inf\{\|x_1 - x_2\|_2 : \ x_1 \in P_1, \ x_2 \in P_2\}$$

can be found using the following quadratic program

$$\begin{aligned}
\min\ & \|x_1 - x_2\|_2^2 \\
& A_1 x_1 \leq b_1 \\
& A_2 x_2 \leq b_2.
\end{aligned}$$

6.7 Exercises

Exercise 6.1 Prove that the second order cone $\mathcal{Q}^n$ is a proper cone.

Exercise 6.2 — More difficult *. Let $f : \mathbb{R}^n \to \mathbb{R}$ be defined as $f(x) := \sum_{i=1}^n |x_i|^{3/2}$. Prove that the function f is convex by representing $f(x) \leq t$ as a set of second order cone constraints.

Hint: show that $t \geq z^{3/2}$, for $t, z \geq 0$ is equivalent to the existence of w such that $wt \geq z^2$ and $z \geq w^2$.

Exercise 6.3 Extend the model in Example 6.6.1 to take transaction costs into account. Let s and b be constants describing the selling and buying fee of a single unit of an asset, respectively. Given an initial portfolio, $x_0 \in \mathbb{R}^n$ and the transaction costs s and b (we only sell and buy assets at the beginning of the period), build a QP that minimizes the risk for a given return r after the end of the period.

Exercise 6.4 Prove that every quadratically constrained quadratic program (QCQP) with positive definite Q_i can be expressed as a Second-order cone program (SOCP). I.e. show that for all symmetric positive definite matrices $Q, Q_1, \dots Q_{m_1} \in \mathcal{S}^n_{++}$, $c \in \mathbb{R}^n$, $p_i \in \mathbb{R}^n, r_i \in \mathbb{R}$ for $i \in [m_1]$, $C \in \mathbb{R}^{m \times n}$, $d \in \mathbb{R}^m$, $A \in \mathbb{R}^{\ell_1 \times n}$ and $b \in \mathbb{R}^{\ell_1}$ there exists $f \in \mathbb{R}^n$, $A_i \in \mathbb{R}^{n_i \times n}$, $b_i \in \mathbb{R}^{n_i}$, $c_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, for $i \in [m_2]$ and $F \in \mathbb{R}^{p_2 \times n}$, $g \in \mathbb{R}^{p_2}$ such that the minimizer of

$$\begin{aligned} &\min\ f^\top x && \text{(SOCP)} \\ &\quad \|A_i x + b_i\|_2 \leq c_i^\top x + d_i \qquad \text{for } i \in [m] \\ &\quad Fx = g \end{aligned}$$

let's you determine the minimum of

$$\begin{aligned} &\min\ x^\top Q x + c^\top x && \text{(QCQP)} \\ &\quad x^\top Q_i x + 2 p_i^\top x + r_i \leq 0 \qquad \text{for } i \in [m] \\ &\quad Cx \leq d \\ &\quad Ax = b. \end{aligned}$$

Optional: Proof the above statement also for positive semidefinite Q_i. Further give an instance of (SOCP) that cannot be solved by solving an instance of (QCQP), proving (QCQP)$\subsetneq$(SOCP).

Exercise 6.5 A well known tool in statistics is the so called least absolute shrinkage and selection operator (LASSO). It is used to perform linear regression when it is assumed that only few of the variables actually have an influence, i.e. in the sparse regime. Technically it is defined for $Y \in \mathbb{R}^n, X \in \mathbb{R}^{n \times p}$ and a penalty parameter $\lambda \geq 0$. Then the *Lasso Estimator* is given by

$$\hat{\beta}(\lambda) := \operatorname{argmin}_{\beta \in \mathbb{R}^p} \left(\frac{1}{n} \|Y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right). \tag{P}$$

i) Write the Lasso minimization (min instead of argmin) as a SOCP.

For stability reasons one can also look at the variation

$$\hat{\beta}(\lambda) := \operatorname{argmin}_{\beta \in \mathbb{R}^p} \left(\frac{1}{n} \|Y - X\beta\|_2 + \lambda \|\beta\|_\alpha^\alpha \right), \tag{P_α}$$

where $\alpha = \frac{r}{q} \geq 1$ for natural r, q.

ii) Write the modified Lasso minimization (P_α) as an SOCP.

Exercise 6.6 — Programming Exercise. Consider the following situation: We want to solve an LP of the form

$$\begin{aligned} \min\ & c^\top x \\ & a_i^\top x \leq b_i \\ & x \geq 0 \end{aligned} \tag{LP}$$

but we only know the constraints a_i with some uncertainty. To be precise we know for each i a radius r_i and that a_i lies in the ball with radius r_i around some center point v_i. We want to find a solution x that is feasible for (LP) regardless of which values the a_i actually attain within their respective ball.

Formulate this problem as a convex optimization problem and implement it in python. You can find example instances on the webpage.

Exercise 6.7 — Programming Exercise. In this exercise you play the role of an investor that wants to invest in cryptocurrencies. You want to invest your capital such that your expected return exceeds some threshhold, while the variance (and thus the risk) is minimal. For this you shall implement a Markowitz portfolio as introduced in Example 6.6.1.

On the webpage you can find a notebook with routines to fetch historic data for several cryptocurrencies. Use part of the data you get there and CVXPY to compute a portfolio, then test its performance on the remaining data. More details are provided in the notebook.

6.8 Solutions

Solution to the Exercise 6.1

By Theorem 4.1.2, $\mathscr{Q}^n$ is self dual, which implies $\mathscr{Q}^n = (\mathscr{Q}^n)^{**}$. By Proposition 1.3.4, this implies that $\mathscr{Q}^n$ is closed and convex.

Note that for $z = (x,c) \in \mathscr{Q}^n$, we always have $c > 0$ unless $z = 0$. This means that $\mathscr{Q}^n$ is pointed. Indeed, if $\pm z \in \mathscr{Q}^n \setminus \{0\}$, we have $c, -c > 0$ which is a contradiction.

Lastly, since $\mathscr{Q}^n = (\mathscr{Q}^n)^*$ and $\mathscr{Q}^n$ is pointed, $\mathscr{Q}^n$ is also solid by Proposition 1.3.4.

Solution to the Exercise 6.2

The goal of this exercise is to show that the epigraph of f defined as

$$\operatorname{epi} f := \left\{ (x,t) \in \mathbb{R}^{n+1} : \ f(x) \leq t \right\}$$

is a convex set, since this implies that f is a convex function (see Exercise 3.2).

We first prove the statement of the hint. Let $t, z \geq 0$. If $t \geq z^{3/2}$, then the substitution $w := \sqrt{z}$ clearly implies $z \geq w^2$. Furthermore, since $t \geq z^{3/2} = w^{-1} z^2$ (for $z > 0$, the case $z = 0$ is trivial), we also have $wt \geq z^2$.

For the opposite implication, we have $wt \geq z^2 \geq w^4$ and thus $t \geq w^3$. This gives the bound $t^4 \geq w^3 t^3 \geq z^6$ and finally $t = \sqrt[4]{t^4} \geq \sqrt[4]{z^6} = z^{3/2}$.

Next, we set up a family of equalities and inequalities that has a solution which includes the variables $(x,t) \in \mathbb{R}^{n+1}$ if and only if (x,t) is a solution of the inequality $f(x) \leq t$. For this, we introduce further vectors of variables $s, w, z \in \mathbb{R}^n$ and define $\mathscr{B}$ as the set of variables $(x,s,w,z,t) \in \mathbb{R}^{4n+1}$ that obey the constraints

$$\begin{aligned} z_i^2 &\leq w_i s_i, \\ w_i^2 &\leq z_i, \\ -z_i \leq x_i &\leq z_i, \\ s_i &\geq 0, \\ w_i &\geq 0 \quad \text{for all } i \in [n]; \\ t &= \sum_{j=1}^n s_j. \end{aligned}$$

Note that $z_i \geq 0$ is implied by the inequality $-z_i \leq x_i \leq z_i$. We must verify that $f(x) \leq t$ for every $(x,s,w,z,t) \in \mathscr{B}$ and that conversely, if $f(x) \leq t$ for any $(x,t) \in \mathbb{R}^{n+1}$, then there exists $(s,w,z) \in \mathbb{R}^{3n}$ such that $(x,s,w,z,t) \in \mathscr{B}$. To prove the first part, we know that for every $i \in [n]$, the constraints above imply $s_i \geq z_i^{3/2}$ by the statement of the hint, as well as $z_i \geq |x_i|$. By combining these results with the definition of t, we see that the inequalities

$$f(x) = \sum_{i=1}^n |x_i|^{3/2} \leq \sum_{i=1}^n z_i^{3/2} \leq \sum_{i=1}^n s_i = t.$$

are valid for every choice of $(x,s,w,z,t) \in \mathscr{B}$. For the opposite direction, if $f(x) \leq t$ for some $(x,t) \in \mathbb{R}^{n+1}$, let us assign the values of z as $z_i := |x_i|$ for every $i \in [n]$ and the values of s as $s_i := |x_i|^{3/2} = z_i^{3/2}$ for $i \in [n-1]$ and $s_n := t - \sum_{i=1}^{n-1} s_i = t - f(x) + z_n^{3/2} \geq z_n^{3/2}$. By this choice of z and s, we $s_i \geq z_n^{3/2}$ for every $i \in [n]$, which means that there exists a vector $w \in \mathbb{R}^n_{\geq 0}$ such that $z_i^2 \leq w_i s_i$ and $w_i^2 \leq z_i$ for every $i \in [n]$ by the statement of the hint. The choice and the existence of the vectors z, s and w respectively provide a solution $(x,s,w,z,t) \in \mathscr{B}$.

We now need to show that all constraints above are (equivalent to) second-order constraints, since this means that $\mathscr{B}$ is the feasibility set of an SOCP and thus a convex set. While it is clear that

linear constraint can be formulated as a second-order constraint, we need to analyze the inequalities that include quadratic terms. For every $i \in [n]$,

$$(z_i, w_i, 2^{-1}s_i) \in \mathscr{Q}^1_{rot} \quad \Leftrightarrow \quad \begin{cases} z_i^2 \leq w_i s_i; \\ s_i, w_i \geq 0, \end{cases}$$

as well as

$$(w_i, z_i, 2^{-1}) \in \mathscr{Q}^1_{rot} \quad \Leftrightarrow \quad \begin{cases} w_i^2 \leq z_i; \\ z_i \geq 0. \end{cases}$$

By Theorem 4.1.1, this means that the inequalities $z_i^2 \leq w_i s_i$ and $w_i^2 \leq z_i$ can be reformulated as second-order constraints, which confirms our claim.

By putting together all the arguments made above ,we know that $\mathscr{B}$ is a convex set and that

$$\text{epi} f = \left\{ (x,t) \in \mathbb{R}^{n+1} : \exists (s,w,z) \in \mathbb{R}^{3n} \text{ s.t. } (x,s,w,z,t) \in \mathscr{B} \right\}.$$

In other words, the epigraph of the function f is a linear projection of the convex set $\mathscr{B}$ and thus a convex set as well, which implies that f is a convex function.

Solution to the Exercise 6.3

Let $s \geq 0$ be the selling fee for each asset unit and $b \geq 0$ be the buying fee for each asset unit. We introduce the variables $u_+, u_- \in \mathbb{R}^n_{\geq 0}$ which correspond to the new investigations and sales of assets. Since $x_0 \in \mathbb{R}^n_{\geq 0}$ denotes the initial portfolio, the end portfolio is given as $x = x_0 + u_+ - u_-$.

We can discard the budget constraint $\frac{1}{B}\mathbf{1}^T x = 1$, but we must make sure that the expenses are equal to the sales, fees included. This is guaranteed by the constraint

$$(1-s)\mathbf{1}^T u_- = (1+b)\mathbf{1}^T u_+.$$

To summarize, the new optimization problem can be formulated as

$$\begin{aligned} \min\ & x^T \Sigma x \\ & \overline{p}^T x \geq r \\ & x = x_0 + u_+ - u_- \\ & (1-s)\mathbf{1}^T u_- = (1+b)\mathbf{1}^T u_+ \\ & x, u_+, u_- \geq 0. \end{aligned}$$

Solution to the Exercise 6.4

We start with given

$$\begin{aligned} & Q, Q_1, \dots Q_{m_1} \in \mathscr{S}^n_{++}, c \in \mathbb{R}^n, \\ & p_i \in \mathbb{R}^n, r_i \in \mathbb{R} \text{ for } i \in [m_1], \\ & C \in \mathbb{R}^{\ell \times n}, d \in \mathbb{R}^\ell \text{ and} \\ & A \in \mathbb{R}^{p \times n}, b \in \mathbb{R}^p. \end{aligned}$$

We immediately have the correspondence $F = A$ and $g = b$. In particular $p_2 = p_1$. Further notice that we can write the inequality constraint $Cx \leq d$ equivalently as

$$c_i^\top x = x^\top 0 x + 2\frac{1}{2} c_i^\top x + 0 \leq 0 \text{ for } i \in [\ell],$$

where c_i denotes the i-th row of C. Therefore it suffices to show the claim for problems of the form

$$\begin{aligned} &\min x^\top Q x + c^\top x && \text{(QCQP)}\\ &\quad x^\top Q_i x + 2p_i^\top x + r_i \le 0 \qquad \text{for } i \in [m_1 + \ell]. \end{aligned}$$

Next we use that Q_i is positive definite and hence we get an invertible square root $Q_i^{\frac{1}{2}}$. This allows us to write

$$x^\top R x + 2a^\top x + r = \left\| R^{\frac{1}{2}} x + R^{-\frac{1}{2}} a \right\|_2^2 + r - a^\top R^{-1} a$$

and hence we can write (QCQP) as

$$\begin{aligned} &\min_{x,t} t && \text{(SOCP)}\\ &\quad \left\| Q_0^{\frac{1}{2}} x + \frac{1}{2} Q_0^{-\frac{1}{2}} c \right\|_2 \le t\\ &\quad \left\| Q_i^{\frac{1}{2}} x + Q_i^{-\frac{1}{2}} p_i \right\|_2 \le \sqrt{p_i^\top Q_i p_i - r_i} \qquad \text{for } i \in [m_1 + \ell]. \end{aligned}$$

The solutions x are the same in this case and we can reconstruct the minimum by either plugging in x or going backwards from the solution t^* and hence get the optimal solution

$$(t^*)^2 - \frac{1}{4} c^\top Q_0^{-\frac{1}{2}} c.$$

of (QCQP). Summing up, the given QCQP is equivalent to the SOCP defined by

$$\begin{aligned} f &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \in \mathbb{R}^{n+1}\\ A_i &= \begin{pmatrix} Q_i^{\frac{1}{2}} \\ 0 \end{pmatrix} \in \mathbb{R}^{(n+1)\times n} && \text{for } i \in \{0, \dots, m_2\}\\ b_0 &= \begin{pmatrix} \frac{1}{2} Q_0^{-\frac{1}{2}} c \\ 0 \end{pmatrix} \in \mathbb{R}^{n+1},\ c_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \in \mathbb{R}^{n+1},\ d_0 = 0\\ b_i &= \begin{pmatrix} Q_i^{-\frac{1}{2}} p_i \\ 0 \end{pmatrix} \in \mathbb{R}^{n+1},\ c_i = 0 \in \mathbb{R}^{n+1},\ d_i = \sqrt{p_i^\top Q_i p_i - r_i} && \text{for } i \in \{1, \dots, m_2\}\\ F &= A,\ g = b, \end{aligned}$$

where we defined $m_2 \coloneqq m_1 + \ell$ and for $i \in \{m+1, \dots, m_2\}$

$$\begin{aligned} Q_i &= 0\\ p_i &= \frac{1}{2} c_i\\ r_i &= 0. \end{aligned}$$

The Optional statement can be proven with the following (different) approach.

1. We start by showing that one can view QCQP as a special case of SOCP. To do so first rewrite QCQP in the following form

$$\begin{aligned} &\min q + c^\top x\\ &\quad x^\top Q x \le q\\ &\quad x^\top Q_i x + p_i^\top x + r_i \le 0 && \text{for } i \in [m]\\ &\quad Cx \le d\\ &\quad Ax = b. \end{aligned}$$

We follow a similar reasoning as for proving that any QP can be captured with SOCP. For every $i \in [m]$ we add a variable $q_i \in \mathbb{R}$ and a constraint

$$q_i = -2p_i^\top x - r_i.$$

The QCQP can be rewritten as:

$$\begin{aligned}
&\min\ q + c^\top x \\
&\quad x^\top Q x \le q \\
&\quad x^\top Q_i x \le q_i && \text{for } i \in [m] \\
&\quad q_i = -2p_i^\top x - r_i && \text{for } i \in [m] \\
&\quad Cx \le d \\
&\quad Ax = b.
\end{aligned}$$

the rest of the proof is analogous to proving that QP can be captured by SOCP. That is, we note that for every $i \in [m]$, $x^\top Q_i x \le q_i$ if and only if $(Q_i^{1/2}x, q_i, 1/2) \in \mathscr{Q}^n_{rot}$ and $x^\top Q x \le q$ if and only if $(Q^{1/2}x, q, 1/2) \in \mathscr{Q}^n_{rot}$.

2. First note that SOCP can be written as QCQP if for every $i \in [m]$, $c_i = 0$, see Definition 6.2.2. If $c_i \neq 0$, the second order cone constraint

$$\|A_i x + b_i\|_2 \le c_i^\top x + d_i$$

cannot be written in the form $q_i(x) \le 0$ for some convex quadratic function q_i. A potential approach is to square the above constraint and obtain the following system containing a quadratic and a linear constraint

$$\|A_i x + b_i\|_2^2 \le \left(c_i^\top x + d_i\right)^2,$$

$$c_i^\top x + d_i \ge 0.$$

However, the first constraint is not convex.

Solution to the Exercise 6.5

We will use Exercise 6.4 and instead write the problem as a QCQP. Therefore we rewrite the penalty term to

$$\begin{aligned}
&\min_{\beta \in \mathbb{R}^p} \left(\frac{1}{n}\|Y - X\beta\|_2^2 + \lambda \mathbb{1}^\top t\right) \\
&\quad s.t.\ t \ge \beta \\
&\qquad t \ge -\beta
\end{aligned}$$

This does already give us a QCQP, finishing *i*).

Now we will use the same idea as in Exercise 6.2 to deal with α from *ii*). Therefore we first proof that inequalities of the form

$$-x^\pi \le t, \tag{6.1}$$

where $0 < \pi = \frac{\ell}{h} \le 1$ and $x \ge 0$, can be expressed as an SOCP constraints. Therefore we first introduce the variable $s \ge 0$, where $s - t \ge 0$, in which case we can equivalently write

$$s - t \le x^\pi, \qquad s \ge 0, \qquad s - t \ge 0.$$

We did this as now both sides of the inequality are positive so taking the h-th power is an equivalent transformation. So we do exactly that:

$$(s-t)^h \leq x^l, \qquad s \geq 0, \qquad s-t \geq 0.$$

Next define $k \coloneqq \lceil \log_2(h) \rceil$ and get the further equivalent formulation

$$(s-t)^{2^k} \leq x^\ell (s-t)^{2^k-h} 1^{h-\ell}, \qquad s \geq 0, \qquad s-t \geq 0.$$

Here we added the ones to get the same number on factors on both sides. Now we will find inequalities in the rotated cone form, i.e. of the form $y_1^2 \leq y_2 y_3$. Therefore first define $(a_i)_{i\in\mathbb{N}_{\geq 0}}$ and $(b_i)_{i\in\mathbb{N}_{\geq 0}}$ as the binary representations of $m \coloneqq 2^k - h$ and ℓ respectively. Now we start by defining the *first level* inequalities

$$w_i^2 \leq (s-t)^{a_i} x^{b_i} \qquad \text{for } i \in \{0,1,\ldots,k-1\}.$$

More intuitively we use x in inequality i if $b_i = 1$ and otherwise replace it with 1 and do the same for $(s-t)$. This will guarantee that we end up with exactly ℓ and m many x respectively $(s-t)$ factors. Next we combine all these first level inequalities step by step with

$$\begin{aligned} v_0^2 &\leq w_0 1 \\ v_j^2 &\leq v_{j-1} w_j \qquad \text{for } j \in \{1,\ldots,k-2\}. \end{aligned}$$

Finally we get back to the original inequality with

$$(s-t)^2 \leq v_{k-2} w_{k-1}.$$

To check if this indeed the correct inequality we plug in and get

$$\begin{aligned} (s-t)^{2^k} &\leq (s-t)^{\Sigma_{i=0}^{k-1} a_i 2^i} x^{\Sigma_{i=0}^{k-1} b_i 2^i} \\ &= (s-t)^m x^\ell. \end{aligned}$$

This method can be seen in Figure 6.1. This finishes claim (6.1).

In the exercise we do however look at inequalities of the form $|\beta_i|^\alpha \leq t$ with $\alpha \geq 1$. In order to use the above this can however easily be modified to the inequalities

$$-t^{\frac{q}{r}} \leq -\beta_i, \qquad -t^{\frac{q}{r}} \leq \beta_i$$

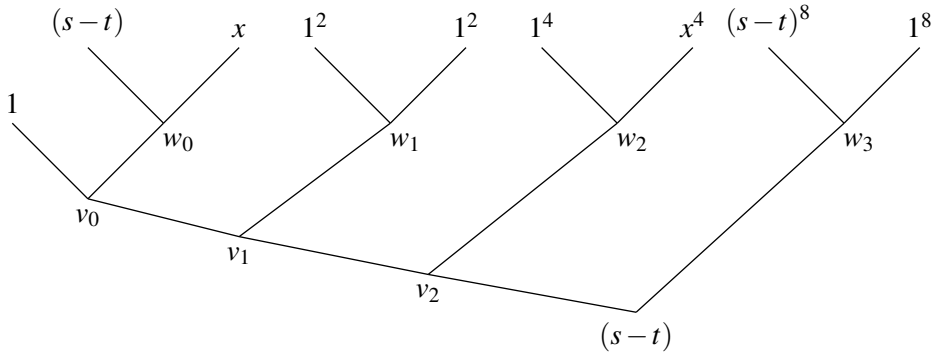
which have exactly the form in (6.1) with $x \leftarrow t, t \leftarrow \pm\beta_i$ and $\pi \leftarrow \frac{1}{\alpha} \leq 1$. Finally we can sum up these inequalities which proofs that we can formulate the inequality

$$\sum_{i=0}^{n} |\beta_i|^{\frac{r}{q}} \leq t$$

as an SOCP constraint. Therefore we get the SOCP

$$\begin{aligned} &\min s + \lambda t \\ &s.t. \ \frac{1}{n} \|Y - X\beta\|_2 \leq s \\ &\quad \|\beta\|_\alpha^\alpha \leq t \end{aligned}$$

Figure 6.1: Example of the SOCP inequality construction for $m = 9$ and $\ell = 5$. In this case the binary representations are $(1,0,0,1)$ and $(1,0,1)$.



7. Semidefinite programs

In this chapter, we study another type of convex optimization problem that belongs to the family of conic programs. Semidefinite programs can be seen as a generalization of linear and second-order cone programs where we optimize a linear objective function subject to linear matrix inequalities (LMI). Before we present semidefinite programs, we introduce basic concepts for positive semidefinite matrices and describe their properties.

7.1 Positive semidefinite cone

Let us start with some basic notation. We denote the set of real n by n symmetric matrices by $\mathcal{S}^n$. There exists a particular subset of real symmetric matrices that deserves special attention, these matrices are called positive semidefinite matrices. They are denoted by $\mathcal{S}^n_+$, and their definition and some basic properties can be found in Appendix 9.1. From now on we will have a more in-depth look at the positive semidefinite cone $\mathcal{S}^n_+$.

The following example visualizes the cone $\mathcal{S}^2_+$ as a subset of $\mathbb{R}^3$.

■ **Example 7.1.1** Consider a matrix $A = \begin{pmatrix} x & z \\ z & y \end{pmatrix}$. Note that $A \in \mathcal{S}^2_+$ if and only if $x \geq 0$, $y \geq 0$ and $xy \geq z^2$. The boundary of the $\mathcal{S}^2_+$ from two different perspectives can be seen in Figure 7.1. ■

We will need two main theorems describing the properties of the PSD cone $\mathcal{S}^n_+$.

Theorem 7.1.2 For every $n \in \mathbb{N}$, the cone $\mathcal{S}^n_+$ of semidefinite matrices is a proper cone.

Proof. First we prove that the cone is convex. Indeed, let $A, B \in \mathcal{S}^n_+$ and let $\mu_1 \geq \ldots \geq \mu_n \geq 0$ and $\phi_1 \geq \ldots \geq \phi_n \geq 0$ be the eigenvalues of A and B, respectively. For any $a \in [0,1]$, by Wely's inequality [23], the smallest eigenvalue of $aA + (1-a)B$ is at least $a\mu_n + (1-a)\phi_n$ which is greater or equal than zero.

Equivalently, by Proposition 9.1.4 we have $\mathcal{S}^n_+ = \{X \in \mathcal{S}^n \mid z^\top X z \geq 0 \text{ for } z \in \mathbb{R}^n\}$. Since, $q_z(X) := z^\top X z$ is a linear continuous function of X, $\mathcal{S}^n_+$ is the intersection of an infinite number of halfspaces, and thus is convex and closed.

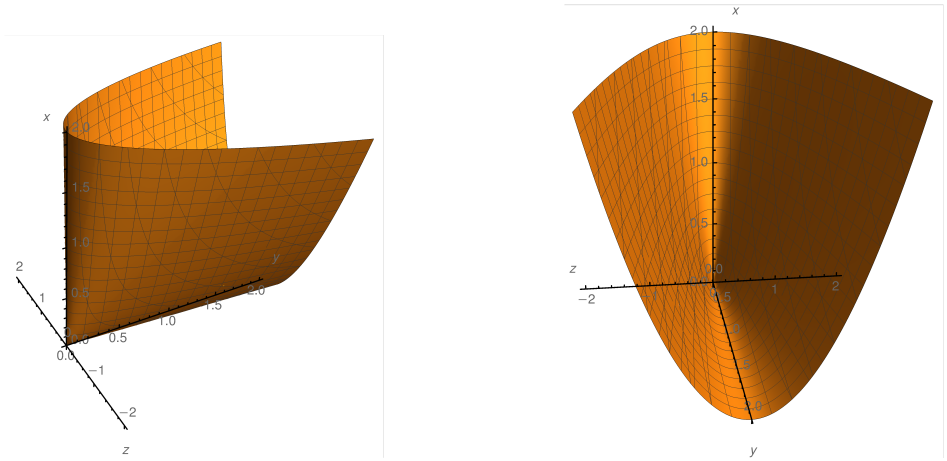


Figure 7.1: Boundary of $\mathscr{S}_+^2$ for the special class of matrices in Example 7.1.1.

Next, we prove that $\mathscr{S}_+^n$ is pointed. For a matrix X if $X, -X \in \mathscr{S}_+^n$ then all the eigenvalues of X have to be zero, thus X is nilpotent. Since $X \in \mathscr{S}_+^n$, by the spectral theorem, X is diagonalizable, and the only diagonalizable nilpotent matrix is the zero matrix.

It remains to show that $\mathscr{S}_+^n$ is solid. Consider the identity matrix $\mathbb{I}_n \in \mathscr{S}_+^n$ and note that the smallest eigenvalue of $\mathbb{I}_n$ is 1. Our goal is to prove that there exist an $\varepsilon > 0$ such that the ball of radius ε around $\mathbb{I}_n$ is in $\mathscr{S}_+^n$. Let $\varepsilon = \frac{1}{2n^2}$, we show that $\mathbb{I}_n - \varepsilon N \in \mathscr{S}_+^n$ for every $N \in \mathscr{S}^n$ for $\|N\|_2 := \sqrt{Tr(N,N)} = 1$. To this end, note that $\sqrt{Tr(N,N)} = 1 \Rightarrow |N_{i,j}| \leq 1$ and thus the maximum eigenvalue of N is at most $\max_{\|x\|_2} x^\top N x \leq n^2$. This in turns implies that the smallest eigenvalue of $\mathbb{I}_n - \varepsilon N$ is at least $1/2$ which completes the proof. ■

In this chapter we frequently use a scalar product for matrices $A, B \in \mathscr{S}^n$ defined as

$$\langle A, B \rangle := Tr\left(A^\top B\right) = Tr\left(AB\right),$$

for $Tr(X)$ being a trace of a matrix $X \in \mathscr{S}^n$ defined as $Tr(X) = \sum_{i=1}^n X_{i,i}$ which implies that $Tr\left(A^\top B\right) = \sum_{i=1}^n \sum_{j=1}^n A_{i,j} B_{i,j}$.

Theorem 7.1.3 For every $n \in \mathbb{N}$, the cone $\mathscr{S}_+^n$ of semidefinite matrices is self dual.

Proof. We start with proving the following three facts:

Fact 1. Every matrix $S \in \mathscr{S}_+^n$ has a PSD *square root* $S^{1/2} \in \mathscr{S}_+^n$, i.e., $S = S^{1/2} S^{1/2}$.

> *Proof.* By Theorem 9.1.5 we know that for every PSD matrix $S \in \mathscr{S}_+^n$ there exists a decomposition of the form $S = Q \Lambda Q^\top$, where Q is a unitary matrix and Λ is a diagonal matrix whose entries are eigenvalues of S.
> Since S is PSD all eigenvalues are nonnegative thus we can define a matrix $\Lambda^{1/2}$ with diagonal entries being square roots of eigenvalues, thus $\Lambda^{1/2} \Lambda^{1/2} = \Lambda$.
> Define $S^{1/2} := Q \Lambda^{1/2} Q^\top$ and note that this matrix is PSD as $\Lambda^{1/2}$ is PSD. Then we have
>
> $$S^{1/2} S^{1/2} = Q \Lambda^{1/2} Q^\top Q \Lambda^{1/2} Q^\top = Q \Lambda^{1/2} \Lambda^{1/2} Q^\top = S.$$
>
> ■

Fact 2. For any matrices $U \in \mathbb{R}^{n \times m}$ and $V \in \mathbb{R}^{m \times n}$ we have

$$Tr(UV) = Tr(VU).$$

Proof. $Tr(UV) = \sum_{i=1}^{n}\sum_{j=1}^{m} U_{i,j}V_{j,i} = \sum_{j=1}^{m}\sum_{i=1}^{n} V_{j,i}U_{i,j} = Tr(VU)$. ■

Fact 3. For any $v \in R^n$, $vv^\top$ is PSD.

Proof. For any $w \in \mathbb{R}^n$, we have $w^\top(vv^\top)w = (v^\top w)^2 \geq 0$. ■

Now we are ready to prove the theorem. Recall that for a cone $K \in \mathbb{R}^n$ the dual cone is defined as $K^* := \{l \in \mathbb{R}^n : \langle l, x\rangle \geq 0, \text{ for all } x \in K\}$.

First we prove that $\mathscr{S}_+^n \subseteq (\mathscr{S}_+^n)^*$. This is equivalent to prove that for every PSD matrices $S, T \in \mathscr{S}_+^n$ we have $\langle S, T\rangle \geq 0$. Indeed,

$$\langle S, T\rangle = Tr(ST) = Tr(S^{1/2}S^{1/2}T) = Tr(S^{1/2}TS^{1/2}),$$

where the second inequality holds by Fact 1, and the third one by Fact 2. Note that $S^{1/2}TS^{1/2} \in \mathscr{S}_+^n$, because $S^{1/2}$ is symmetric. Thus, for every $v \in \mathbb{R}^n$, $v^\top S^{1/2}TS^{1/2}v = (S^{1/2}v)^\top T(S^{1/2}v) \geq 0$. Which implies that its trace is nonnegative (since it is the sum of nonnegative eigenvalues).

Second we prove that $\mathscr{S}_+^n \supseteq (\mathscr{S}_+^n)^*$. Consider a matrix $S \in (\mathscr{S}_+^n)^*$, we prove that $S \in \mathscr{S}_+^n$, that is for every $u \in \mathbb{R}^n$, $u^\top Su \geq 0$. Indeed,

$$u^\top Su = Tr(u^\top Su) = Tr(Suu^\top) = \left\langle S, uu^\top \right\rangle \geq 0,$$

where the second equality comes from Fact 2 and the last inequality comes from the fact that $S \in (\mathscr{S}_+^n)^*$ and $uu^\top \in \mathscr{S}_+^n$. ■

7.2 Spectrahedra and spectrahedral shadows

In this section, we first define spectrahedra, which are a generalization of polyhedra studied in Chapter 1. Then, we define spectrahedral shadows and study their properties.

We start with the a definition of linear matrix inequality that is a direct generalization of linear inequalities studied in Chapter 1 for linear programs.

Definition 7.2.1 — Linear matrix inequality. Let $A_i \in \mathscr{S}^n$ for $i = \{0, 1, \ldots, m\}$ and $x \in \mathbb{R}^m$. A linear matrix inequality (LMI), is an expression of the form:

$$A_0 + \sum_{i=1}^{m} A_i x_i \succeq 0.$$

Definition 7.2.2 — Spectrahedron. A set $S \subseteq \mathbb{R}^m$ is called a *spectrahedron* if it is of the form

$$S = \{x \in \mathbb{R}^m : A_0 + \sum_{i=1}^{m} A_i x_i \succeq 0\},$$

for some matrices $A_i \in \mathscr{S}^n$ for $i = \{0, \ldots, m\}$.

We start with proving that every spectrahedron is a convex set.

Theorem 7.2.3 Every spectrahedron is a convex set.

Proof. First we prove that every spectrahedron is a convex set. Let $A_i \in \mathscr{S}^n$ for $i \in \{0, \ldots, m\}$ and let $x, y \in \mathbb{R}^m$ be such that $A_0 + \sum_{i=1}^m A_i x_i \succeq 0$ and $A_0 + \sum_{i=1}^m A_i y_i \succeq 0$. We prove that for every $\lambda \in [0,1]$ we have $A_0 + \sum_{i=1}^m A_i(\lambda x_i + (1-\lambda)y_i) \succeq 0$. Indeed, $A_0 + \sum_{i=1}^m A_i(\lambda x_i + (1-\lambda)y_i) = \lambda(A_0 + \sum_{i=1}^m A_i x_i) + (1-\lambda)(A_0 + \sum_{i=1}^m A_i y_i)$ which is positive semidefinite, since by Weyl's inequality [23] the smallest eigenvalue of the summation of two matrices is at least the sum of the smallest eigenvalues of those matrices.

■

Definition 7.2.4 — Spectrahedral shadow. A *spectrahedral shadow* is an image of a spectrahedron under a linear projection. That is, a set $T \subseteq \mathbb{R}^n$ is a spectrahedral shadow if there exist a spectrahedron $S \in \mathbb{R}^m$ and a linear projection $\pi : R^m \to \mathbb{R}^n$ such that $\pi(S) = T$.

7.3 Semidefinite program

In this section we define semidefinite programs that are a special subfamily of conic programs.

We adapt Definition 2.2.1 for the case when K is $\mathscr{S}^n_+$.

Definition 7.3.1 Let $A_1, \ldots, A_m, F_1, \ldots, F_n, G, C \in \mathscr{S}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$. A *semidefinite program (SDP)* in the general and in standard form is the following:

$$\begin{array}{ll} \textit{General}: & \textit{Standard}: \\ \min c^\top x & \min \langle C, X \rangle \\ \quad x_1 F_1 + \ldots + x_n F_n + G \preceq 0 & \quad \langle A_i, X \rangle = b_i \qquad \text{for } i = \{1, \ldots m\} \\ \quad Ax = b & \quad X \succeq 0 \end{array}$$

R Note that the feasibility set of an SDP is a convex set. For the general form it follows directly from Theorem 7.2.3. For the standard form, it is an intersection of the proper cone of positive semidefinte matrices, see Theorem 7.1.2, and an affine subspace, both of which are convex sets thus, by Proposition 1.1.4, it is convex.

■ **Example 7.3.2** Consider the following SDP program

$$\begin{aligned} \min\ & x_{11} + 2x_{13} \\ & x_{11} + 2x_{14} + 2x_{23} = 1 \\ & x_{33} + x_{44} = 0 \\ & X \succeq 0 \end{aligned}$$

This is an SPD in a standard form and it can be expressed like in Definition 7.3.1 with the following choice of matrices:

$$C = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad A_1 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \qquad A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Note that the constraint $x_{33} + x_{44} = 0$ implies $x_{33} = 0$ and $x_{44} = 0$. This in turns implies $x_{13} = 0$. Indeed, consider a principal submatrix of X composed of the first and the third row/column. The determinant of this matrix has to be zero since a diagonal element is zero, which implies that the off-diagonal elements have to be zero. Analogous reasoning can be made to show that x_{14} and x_{23} equal to zero by considering a principal submatrix of X composed of the first and the fourth row/column and the second and the third row/column, respectively. This shows that the minimum value is equal to 1. ■

R Semidefinite programs contain linear programs as a special case. To see this consider a linear program of the form

$$\min c^\top x | \; Ax \leq b$$

and note that it is equivalent to the following semidefinite program

$$\min c^\top x | \; \operatorname{diag}\left(a_1^\top x - b_1, \ldots, a_m^\top x - b_m\right) \preceq 0,$$

which can be easily transformed into general form. Note that we can model many linear inequalities with one linear matrix inequality that involves only diagonal matrices.

Theorem 7.3.3 Every semidefinite program in general form can be formulated as a semidefinite program in standard form and vice versa.

Proof. Consider a semidefinite program in general form as in Definition 7.3.1. First, for every variable x_i, for $i \in [n]$ we introduce two nonnegative variables x_i^+, x_i^-, such that $x_i = x_i^+ - x_i^-$. Next, we introduce a slack variable matrix Y such that

$$x_1 F_1 + \ldots + x_n F_n + G + Y = 0.$$

Then the problem becomes

$$\begin{aligned}
\min\ & c^\top x^+ - c^\top x^- \\
& (x_1^+ - x_1^-)F_1 + \ldots + (x_n^+ - x_n^-)F_n + G + Y = 0 \\
& A(x^+ - x^-) = b \\
& x_i^+, x_i^- \geq 0 \qquad \text{for } i \in [n] \\
& Y \succeq 0
\end{aligned}$$

which can be further transformed to standard form where variable matrix X is a block diagonal matrix composed of blocks Y and x_i^+ and x_i^- for every $i \in [n]$ and the objective, and the first, and the second constrain can be written in the matrix inner product form.

To transform a semidefinite program from standard to general form we introduce vectors $c, a^i \in \mathbb{R}^{\binom{n+1}{2}}$ for $i \in [m]$ and a vector of variables x, such that $c_{k,k} = C_{k,k}$, $a^i_{k,k} = (A_i)_{k,k}$ and $x_{k,k} = X_{k,k}$, for $k \in [n]$ and $c_{k,\ell} = 2C_{k,\ell}$, $a^i_{k,\ell} = 2(A_i)_{k,\ell}$ and $x_{k,\ell} = X_{k,\ell}$, for $k, \ell \in [n]$, $k < \ell$. For $k, \ell \in [n]$ let $e^k \in \mathbb{R}^n$ be a unit vector with all zero entries but one in the k'th entry. Then $e^k \left(e^k\right)^\top$ and $\left(e^k \left(e^\ell\right)^\top + e^\ell \left(e^k\right)^\top\right)$ are symmetric metrices and the problem in general form can be written as

$$\begin{aligned}
\min\ & c^\top x \\
& \sum_{k=1}^{n} x_{k,k} e^k \left(e^k\right)^\top + \sum_{k=1}^{n} \sum_{\ell=k+1}^{n} x_{k,\ell} \left(e^k \left(e^\ell\right)^\top + e^\ell \left(e^k\right)^\top\right) \succeq 0 \\
& \left(a^i\right)^\top x = b_i \qquad \text{for } i \in [m]
\end{aligned}$$

■

Next, we prove that every second-order cone program is a special case of a semidefinite program. We do it by showing that every second-order cone constraint (Definition 6.2.1) is a linear matrix inequality (Definition 7.2.1). This implies that the second-order cone program in a general form can be formulated as the semidefinite program in a general form.

Theorem 7.3.4 Every second-order cone constraint is a linear matrix inequality.

Proof. Before we prove the theorem, we want to prove the following intermediate result. Let $x \in \mathbb{R}^n$ and $t \in \mathbb{R}_{\geq 0}$, then

$$\begin{pmatrix} t & x_1 & \ldots & x_n \\ x_1 & t & & 0 \\ \vdots & & \ddots & \\ x_n & 0 & & t \end{pmatrix} \succeq 0 \iff \|x\|_2 \leq t.$$

To prove the fact, note that by the Schur complement (Proposition 9.1.10), we have

$$\begin{pmatrix} t & x_1 & \dots & x_n \\ x_1 & t & & 0 \\ \vdots & & \ddots & \\ x_n & 0 & & t \end{pmatrix} \succeq 0 \iff t\mathbb{I}_n - \frac{1}{t}xx^\top \succeq 0,$$

if $t > 0$. Now, note that adding the matrix $t\mathbb{I}_n$ to any matrix increases all its eigenvalues by t. Thus $t\mathbb{I}_n - \frac{1}{t}xx^\top \succeq 0$ if and only if the largest eigenvalue of $\frac{1}{t}xx^\top$ is at most t. Since $\frac{1}{t}xx^\top$ is a rank one matrix with all eigenvalues zero, but one equal to $\frac{1}{t}x^\top x$, the constraint $t\mathbb{I}_n - \frac{1}{t}xx^\top \succeq 0$ is satisfied if and only if $\frac{1}{t}x^\top x \leq t$ which is equivalent to $\|x\|_2 \leq t$. For the case $t = 0$, Proposition 9.1.12 yields that x must be 0 for the matrix to be PSD, completing the intermediate proof.

Let $A \in \mathbb{R}^{n_0 \times n}$, $b \in \mathbb{R}^{n_0}$, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$. Consider the second-order cone constraint of the form

$$\|Ax+b\|_2 \leq c^\top x + d.$$

By the fact proved above, we have

$$\begin{pmatrix} c^\top x+d & (Ax+b)_1 & \dots & (Ax+b)_n \\ (Ax+b)_1 & c^\top x+d & & 0 \\ \vdots & & \ddots & \\ (Ax+b)_n & 0 & & c^\top x+d \end{pmatrix} \succeq 0 \iff \|Ax+b\|_2 \leq c^\top x + d.$$

The constraint on the left can be expressed as a linear matrix inequality.

Note that, similarly to casting linear constraints as linear matrix inequalities, we can write many different second-order cone constraints as one linear matrix inequalities without adding additional variables just by increasing the size of matrices.

■

7.4 Duality for semidefinite programs

Instead of using Definition 2.2.5 we will derive the dual of semidefinite programs explicitly in this section. Since we proved that every SDP in general form can be written in standard form, we will derive it for this form.

Therefore first note that we cannot directly apply our duality theory to the standard from, since $X \succeq 0$ is not expressed as a scalar function. To remedy this fact we instead use the minimal eigenvalue function of symmetric matrices

$$\begin{aligned} \lambda_{\min}(A) := \min_Y \ & \langle A, Y \rangle \\ s.t. \ & Y \succeq 0 \\ & \operatorname{Tr}(Y) = 1. \end{aligned}$$

This representation can be derived by writing both A and Y in their eigenvalue decomposition and multiplying out. Since $X \succeq 0$ also forces X to be symmetric, we will choose $\mathbb{P} = \mathscr{S}^n$. This allows us to rewrite the problem in standard form to

$$\begin{aligned} \min \ & \langle C, X \rangle \\ & \langle A_i, X \rangle = b_i \qquad \text{for } i = \{1, \dots, m\} \\ & \lambda_{\min}(A) \geq 0. \end{aligned}$$

Note that since $\lambda_{\min}$ is concave, this formulation is still a convex program. This yields the Lagrangian

$$\begin{aligned} L(X,\lambda,\nu) &= \langle C,X\rangle + \sum_{i=1}^{m} \nu_i (\langle A_i,X\rangle - b_i) - \lambda\cdot\lambda_{\min}(X) \\ &= -\nu^\top b + \left\langle C + \sum_{i=1}^{m} \nu_i A_i, X \right\rangle - \lambda\cdot\lambda_{\min}(X). \end{aligned}$$

Now we can calculate the lagrange dual function

$$\widehat{L}(\lambda,\nu) = \inf_{X\in\mathbb{P}} -\nu^\top b + \left\langle C + \sum_{i=1}^{m} \nu_i A_i, X \right\rangle - \lambda\cdot\lambda_{\min}(X)$$

by instead looking for maximizers of the negative, i.e.

$$\sup_{X\in\mathbb{P}} \nu^\top b + \left\langle -C - \sum_{i=1}^{m} \nu_i A_i, X \right\rangle + \lambda\cdot\lambda_{\min}(X)$$

Since the first term is constant with respect to X, we only have to look at

$$G(\lambda,Z) \coloneqq \sup_{X\in\mathbb{P}} (\langle Z,X\rangle + \lambda\cdot\lambda_{\min}(X))$$

where $Z = -C - \sum_{i=1}^{m} \nu_i A_i$. We first suppose $\lambda \geq 0$ and calculate

$$\begin{aligned} G(\lambda,Z) &= \sup_{X\in\mathbb{P}} \left(\langle Z,X\rangle + \lambda \min_{Y\succeq 0,\ \mathrm{Tr}Y=1} \langle Y,X\rangle \right) \\ &= \sup_{X\in\mathbb{P}} \min_{Y\succeq 0,\ \mathrm{Tr}Y=1} \langle Z+\lambda Y,X\rangle \\ &= \sup_{X\in\mathbb{P}} \min_{Y\succeq 0,\ \mathrm{Tr}Y=\lambda} \langle Z+Y,X\rangle \\ &\overset{3.4}{\leq} \min_{Y\succeq 0,\ \mathrm{Tr}Y=\lambda} \sup_{X\in\mathbb{P}} \langle Z+Y,X\rangle \\ &= \min_{Y\succeq 0,\ \mathrm{Tr}Y=\lambda} \begin{cases} 0, & \text{if } Z+Y=0 \\ \infty, & \text{otherwise.} \end{cases} \\ &= \underline{G}(\lambda,Z), \end{aligned}$$

where

$$\underline{G}(\lambda,Z) \coloneqq \begin{cases} 0, & \text{if } \mathrm{Tr}(Z) = -\lambda \leq 0 \text{ and } Z \preceq 0 \\ \infty, & \text{otherwise.} \end{cases}$$

We will now show that $\underline{G} = G$ (even if $\lambda < 0$). First note that $G \geq 0$ by choosing $X = 0 \in \mathbb{P}$. Hence for $\mathrm{Tr}Z = -\lambda \leq 0$ and $Z \preceq 0$

$$0 \leq G(\lambda,Z) \leq \underline{G}(\lambda,Z) = 0$$

which implies equality for this class of matrices. Now we show that for every other case, G also takes the value infinity. Therefore we will look at 3 different cases.

1) Suppose $\mathrm{Tr}(Z) \neq -\lambda$. Then we can choose $X = stI_n$ where $s = \mathrm{sign}(\mathrm{Tr}(Z)+\lambda) \neq 0$ and get

$$\begin{aligned}
&\langle Z, X\rangle + \lambda \min_{Y\succeq 0,\ \mathrm{Tr}Y=1} \langle Y, X\rangle \\
&= \left\langle Z + \frac{\lambda}{n} I_n, X \right\rangle - \left\langle \frac{\lambda}{n} I_n, X \right\rangle + \lambda \lambda_{\min}(X) \\
&= t\,|Tr(Z)+\lambda| - ts\lambda + ts\lambda \\
&= t\,|Tr(Z)+\lambda| \overset{t\to\infty}{\to} \infty.
\end{aligned}$$

This shows $G(\lambda, Z) = \infty$ in this case.

2) Suppose $\mathrm{Tr}(Z) = -\lambda \leq 0$ but $\lambda_{\max}(Z) > 0$. Then let u be a unit length eigenvector of the largest eigenvalue of Z and $X = tuu^\top$. Then we again get

$$\begin{aligned}
&\langle Z, X\rangle + \lambda \min_{Y\succeq 0,\ \mathrm{Tr}Y=1} \langle Y, X\rangle \\
&= t\lambda_{\max}(Z) \overset{t\to\infty}{\to} \infty
\end{aligned}$$

3) Suppose $Tr(Z) = -\lambda > 0$. Since the trace of Z is the sum of its eigenvalues, there must be at least one positive eigenvalue and hence we can use the same arguments as in 2) to show $G(\lambda, Z) = \infty$.

Now we finally are able to combine the previous results which proved

$$\widehat{L}(\lambda, \nu) = \begin{cases} -\nu^\top b, & \text{if } \mathrm{Tr}(Z) = -\lambda \leq 0 \text{ and } Z \preceq 0 \\ -\infty, & \text{otherwise,} \end{cases}$$

where $Z = -C - \sum_{i=1}^m \nu_i A_i$.

The dual problem can now be written as

$$\begin{aligned}
\max_{\lambda,\nu}\ & -\nu^\top b \\
\text{s.t. } & -C - \sum_{i=1}^m \nu_i A_i \preceq 0 \\
& \mathrm{Tr}\left(-C - \sum_{i=1}^m \nu_i A_i\right) = -\lambda \leq 0,
\end{aligned}$$

which can be transformed to

$$\begin{aligned}
\max_{\lambda,\nu}\ & \nu^\top b \\
\text{s.t. } & \sum_{i=1}^m \nu_i A_i \preceq C \\
& \mathrm{Tr}\left(-C + \sum_{i=1}^m \nu_i A_i\right) = -\lambda \leq 0
\end{aligned}$$

by setting $\nu \leftarrow -\nu$. Since $Z = -C + \sum_{i=1}^m \nu_i A_i \preceq 0$ does already imply $\mathrm{Tr}(Z) \leq 0$ and λ can take any nonnegative value, we can remove the second constraint and end up with the dual problem

$$\begin{aligned}
\max_{\nu}\ & \nu^\top b \\
\text{s.t. } & \sum_{i=1}^m \nu_i A_i \preceq C.
\end{aligned}$$

Therefore we derived the following theorem.

Theorem 7.4.1 — semidefinite program duality. Let $A_1,\ldots,A_m,C \in \mathscr{S}^n$ and $b \in \mathbb{R}^m$. The *primal* and the *dual* formulations of a semidefinite program in standard form are the following.

$$\begin{array}{ll} \textit{Primal}: & \textit{Dual}: \\ \min \ \langle C,X\rangle & \max \ \langle y,b\rangle \\ \quad \langle A_i,X\rangle = b_i \qquad \text{for } i=\{1,\ldots,m\} & \quad \sum_{i=1}^m A_i y_i \preceq C \\ \quad X \succeq 0 & \end{array}$$

Note that by self duality this result is the same as if we would have used Definition 2.2.5.

As for general conic programs weak duality holds, for SDPs it can be derived from Theorem 2.2.6 in the following way:

$$\langle C,X\rangle - b^\top y = \langle C,X\rangle - \sum_{i=1}^m y_i \langle A_i,X\rangle = \left\langle C - \sum_{i=1}^m y_i A_i, X \right\rangle \geq 0 \tag{7.1}$$

However, unlike for linear programs, strong duality does not always holds for SDP, as shown in the following example.

■ **Example 7.4.2** Consider the SDP program from Example 7.3.2. By Theorem 7.4.1, its dual formulation takes the following form:

$$\max \ y_1$$
$$\begin{pmatrix} y_1 & 0 & 0 & y_1 \\ 0 & 0 & y_1 & 0 \\ 0 & y_1 & y_2 & 0 \\ y_1 & 0 & 0 & y_2 \end{pmatrix} \preceq \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The PSD constraint is equivalent to

$$\begin{pmatrix} 1-y_1 & 0 & 1 & -y_1 \\ 0 & 0 & -y_1 & 0 \\ 1 & -y_1 & -y_2 & 0 \\ -y_1 & 0 & 0 & -y_2 \end{pmatrix} \succeq 0$$

which implies that y_1 has to be 0. Note that the primal formulation took optimal value 1, see Example 7.3.2.

■

The primal-dual pair of SDP programs shown in Example 7.3.2 and 7.4.2 is pathological in the sense that every feasible solution does not satisfy the constraint $X \succ 0$. Indeed, we force two diagonal variables x_{33} and x_{44} to be zero in the primal formulation, thus the matrix X has eigenvalue 0 (determinant, which is a product of eigenvalues, is zero; another way to prove it, is to use Gershgorin circle theorem applied to diagonal entries equal to zero). Similarly, in the dual formulation the matrix $C - y_1A_1 - y_2A_2$ has eigenvalue 0 and thus does not satisfy the constraint $C - \sum_{i=1}^n y_i A_i \succ 0$. The requirement for the primal and the dual SDP to be strictly feasible is known as *Slater's condition*, i.e., that there exist solutions satisfying $x \succ 0$ and $C - y_1A_1 - y_2A_2 \succ 0$ for the primal and the dual formulation, respectively.

Theorem 7.4.3 If both the primal and the dual SDP programs are strictly feasible then their

optimal values are the same.

Since the strong duality does not always hold for semidefinite programs, the analogue of complementary slackness from linear programs also does not hold. Nevertheless, the following variant is true.

Theorem 7.4.4 — complementary slackness for SDP. Let X, y be feasible primal and dual solutions for a semidefinite program in standard form. If they satisfy

$$\left\langle C - \sum_{i=1}^{m} A_i y_i, X \right\rangle = 0$$

then X and y are optimal solutions and have the same value.

Proof. The proof directly follows from the weak duality in Equation 7.1 which implies that

$$\langle C, X \rangle = b^\top y$$

■

In Example 7.4.2 we presented a feasible primal and a feasible dual solution for a semidefinite program such that their values were not the same. That showed that the *duality gap* might be non-zero for semidefnite programs. In the following example we show that the situation can be even more complicated, namely we show an example of the primal and the dual semidefinite programs that have no duality gap, but the optimal solution might be never attained.

■ **Example 7.4.5** Consider the following pair of the primal and the dual semidefinite programs.

$$\begin{aligned} &\textit{Primal}: && \textit{Dual}: \\ &\min x_{11} && \max y \\ &2x_{12} = 1 && \begin{pmatrix} 0 & y \\ y & 0 \end{pmatrix} \preceq \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \\ &\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \succeq 0 \end{aligned}$$

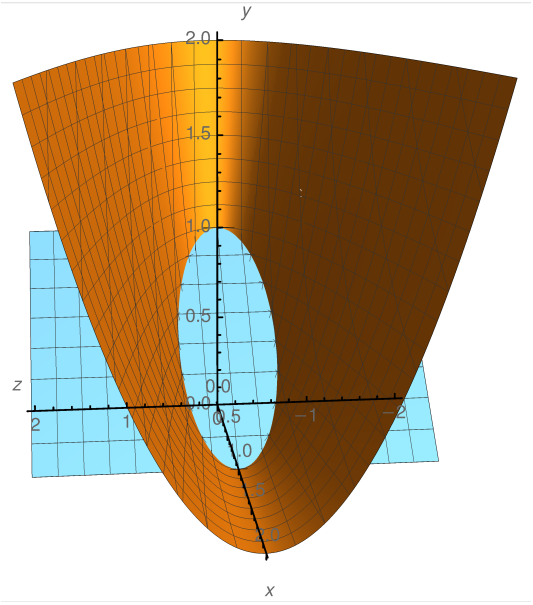
Note that for the dual program, the optimal solution is $y = 0$ however, for the primal x_{11} cannot be zero as it violates the linear matrix inequality. Moreover, for any $\varepsilon > 0$, the solution $x_{11} = \varepsilon$ and $x_{22} = 1/\varepsilon$ is a feasible solution with objective value ε which shows that there is no duality gap. Still, the primal solution of the value zero is never attained. ■

In Example 7.1.1 we saw the boundary of the cone $\mathscr{S}^2_+$ as a subset of $\mathbb{R}^3$. In the following, we give more examples to visualize the concept of spectrahedra.

■ **Example 7.4.6 — Spectraplex.** Consider an intersection of the cone of semidefinite matrices $\mathscr{S}^n_+$ with the affine subspace of matrices with trace 1:

$$\mathscr{O}_n := \{X \in \mathscr{S}^n_+ : Tr(X) = 1\}.$$

This set is called the *spectraplex*. The spectraplex for $n = 2$ can be seen below



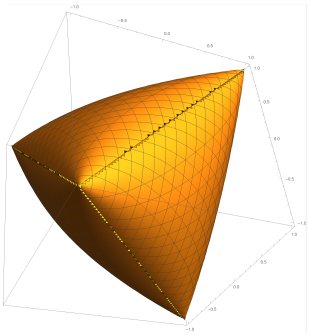
■

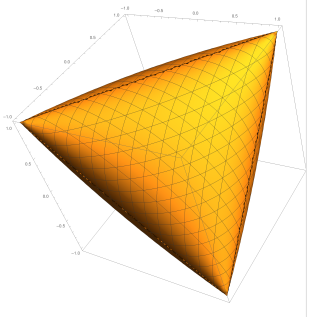
Another interesting and widely presented example is the *elliptope*.

■ **Example 7.4.7 — Elliptope.** Consider the set

$$\mathscr{E}_n := \{X \in \mathscr{S}_+^n : X_{ii} = 1 \text{ for } i = \{1,\ldots,n\}\}.$$

Below one can see the elliptope for $n = 3$ from different perspectives.





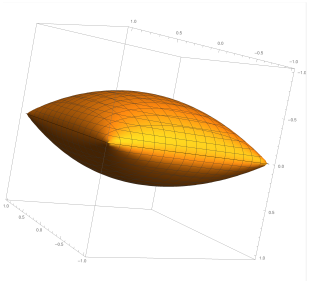
■

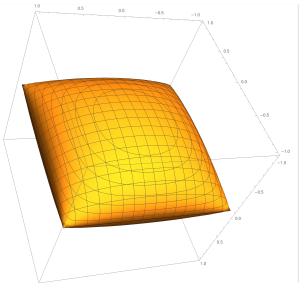
Here is another example of a spectrahedron.

■ **Example 7.4.8** Consider the set

$$S := \{(x,y,z) \in \mathbb{R}^3 : \begin{pmatrix} 1 & x & 0 & 0 \\ x & 1 & y & 0 \\ 0 & y & 1 & z \\ 0 & 0 & z & 1 \end{pmatrix} \succeq 0\}.$$

That is visualized below.





■

7.5 Applications

In this section, we show applications of semidefinite programs to solve optimization problems. We start with a seminal result by Goemans and Williamson for the Max-Cut problem [8].

Definition 7.5.1 — Max-Cut problem.
Input: A graph $G(V,E)$ with n vertices, i.e., $|V| = n$ and a vector of weights w_e for $e \in E$ such that $\sum_{e\in E} w_e = 1$.
Output: A subset of vertices $S \subseteq V$ such that the cut, i.e., weighted sum of edges, between S and $V \setminus S$, denoted by $(S, V \setminus S)$, is maximized.

First, note that a very natural variant of the problem is when every edge has weight one, then the size of a maximum cut is equal to the number of edges in the cut. This problem does not fall in the above definition however, the vector of weighs can be easily scaled such that all the requirements of Definition 7.5.1 are satisfied.

An important fact is that solving the Max-Cut problem in general graphs is NP-hard even in the case when all the weights are the same. In this book, we do not provide proof of this fact, but interested readers can find the proof in [7]. In this section, we focus on designing efficient approximation algorithms based on mathematical programming methods. We start with noting that for some classes of graphs, the problem is actually very easy, this includes the following examples:

- G is bipartite: an optimal solution is one side of the bipartition, and the size of the maximum cut is 1,
- G is complete, $w_e = 1/|E|$ for $e \in E$: an optimal solution is any subset of $\lceil n/2 \rceil$ vertices and the size of the maximum cut is $\frac{1}{2} + \frac{1}{2(n-1)}$ for even n and $\frac{1}{2} + \frac{1}{2n}$ for odd n.

Moreover, there exists a lower bound on the size of the maximum cut, valid for any graph G, that is formalized in the following theorem.

Theorem 7.5.2 For any graph G and a weight vector w_e for $e \in E$ such that $\sum_{e\in E} w_e = 1$ there exists a subset of vertices $S \subseteq V$ such that the size of the cut $(S, V \setminus S)$ is at least $1/2$.

Proof. The proof is algorithmic. We show that if S is chosen at random (every vertex $v \in V$ belongs to S with probability $1/2$), the expected size of the cut is $1/2$.

$$\begin{aligned}
\mathbb{E}[|(S, V \setminus S)|] &= \sum_{(u,v)\in E} w_{u,v} \Pr[(u \in S \wedge v \in V \setminus S) \vee (u \in V \setminus S \wedge v \in S)] \\
&= \sum_{(u,v)\in E} w_{u,v} \frac{1}{2} \\
&= \frac{1}{2}.
\end{aligned}$$

■

Note that the proof of Theorem 7.5.2 gives an actual algorithm to solve the Max-Cut problem. Moreover, in expectation the algorithm is a 2 approximation with high probability.

Linear programming formulation

Now, we make an attempt to construct an algorithm based on the linear programming techniques that we studied in the first chapter. Of course we do not hope to solve the problem exactly, as it is NP-hard, but we aim for efficient approximation algorithms. We start with a construction of a $0/1$ integer program that we relax in the next step. To this end we need two vectors of $0/1$ variables $z \in \{0,1\}^{|E|}$ and $y \in \{0,1\}^{|V|}$. The intended meaning of the variable y_v is to take value 1 is $v \in S$ and z_e is to take value 1 if the edge e is in the cut and zero otherwise. We consider the following program.

$$\begin{aligned}
\max \sum_{(u,v)\in E} & w_{u,v} z_{u,v} \\
& z_{u,v} \leq y_u + y_v && \text{for } (u,v) \in E \\
& z_{u,v} \leq 2 - (y_u + y_v) && \text{for } (u,v) \in E \\
& z \in \{0,1\}^{|E|} \\
& y \in \{0,1\}^{|V|}
\end{aligned}$$

It is easy to check that an optimal solution (z^*, y^*) corresponds to a maximum cut in a graph $(S, V \setminus S)$, where the set S can be constructed in the following way $S = \{v \in V : \ y_v = 1\}$.

Since the above $0/1$ integer program cannot be solved efficiently for general graphs, we consider the linear relaxation obtained by replacing $0/1$ constraints on variable y and z with

$$\begin{aligned}
& z \in [0,1]^{|E|} \\
& y \in [0,1]^{|V|}
\end{aligned}$$

An optimal solution for this linear program can be found efficiently, and the goal is to analyze its performance. The quality of relaxation is usually quantified with an *integrality gap* measure. The integrality gap of a linear program LP that is a relaxation of the integer program IP is a maximum ratio over feasible instances of an optimal LP solution and an optimal IP. If the integrality gap is equal to 1, we say that relaxation is *exact* and it is a highly desirable situation. In comparison, if the integrality gap is equal to $p > 1$, it is not possible to construct a better than p approximation algorithm using an optimal LP solution as a bound on the optimal value.

Theorem 7.5.3 The integrality gap of the constructed linear program is at least $2\frac{n}{n+1}$.

Proof. Note that the size of a maximum cut in a complete graph K_n, for odd n, is $\frac{1}{2} + \frac{1}{2n}$. However, a solution that puts $y_v = 1/2$ for every $v \in V$ and $z_e = 1$ for every edge $e \in E$ is a feasible solution with an objective value equal to 1. This implies that the integrality gap is at least $2\frac{n}{n+1}$ ■

Note that an integrality gap equal to $2\frac{n}{n+1}$ excludes the existence of a better than $2\frac{n}{n+1}$-approximation algorithm when using a solution of the constructed linear program as an upper bound on the size of an optimal solution. Asymptotically, for $n \to \infty$ the integrality gap is 2 which provides no better algorithm than the trivial algorithm based on the random cut presented in the proof of Theorem 7.5.2.

One could argue that a reason for the poor performance of the linear programming-based approach is the wrong choice of the linear program formulation. Indeed, there exist many different ways to model the feasibility set of the Max-Cut problem as a $0/1$ integer program that might lead

to other linear program relaxations. Choosing a good relaxation is not an easy task, and, at least potentially, a better choice could provide a better algorithm based on linear programming techniques. However, as proved recently in a sequence of results by Charikar, Makarychev, Makarychev [4] and Chan, Lee, Raghavendra, Steurer [3] choosing a better linear programming relaxation is not possible. It is informally summarized in the following theorem.

Theorem 7.5.4 — Charikar, Makarychev, Makarychev 2009 (4) and Chan, Lee, Raghavendra, Steurer 2016 (3)—Informally. For any $\varepsilon > 0$ there does not exist a linear program of polynomial size with an integrality gap for the Max-Cut problem smaller than $2 - \varepsilon$.

Semidefinite programming formulation

We attempt to construct an algorithm based on the semidefinite programming techniques we studied in this chapter. Similar to the previous paragraph, we start modeling the Max-Cut problem as a binary integer program. However, for the sake of presentation, we work now with $-1/1$ variables instead of $0/1$ variables. We use a vector of variables $x \in \{-1,1\}^{|V|}$ with the intended meaning that $x_v = 1$ if $v \in S$, then for $u \in S$ and $v \in V \setminus S$, $\left(\frac{1-x_u x_v}{2}\right) = 1$. We construct the following program

$$\begin{aligned} \max \sum_{(u,v)\in E} w_{u,v} \left(\frac{1-x_u x_v}{2}\right) \\ x \in \{-1,1\}^n \end{aligned}$$

First note that the objective is a quadratic function, however the problem can be also be expressed equivalently in the following way

$$\begin{aligned} \max\ & x^\top W x \\ & x \in \{-1,1\}^n, \end{aligned} \tag{7.2}$$

for some symmetric matrix $W \in \mathscr{S}^n$. Which in turns can be written as

$$\begin{aligned} \max\ & Tr(WX) \\ & X \succeq 0 \\ & X_{i,i} = 1 \qquad \text{for } i \in [n] \\ & \operatorname{rank}(X) = 1 \end{aligned} \tag{7.3}$$

where $X \in \mathscr{S}^n$ is a variable matrix, and W is $1/4$ times the Laplacian of the graph G with entries weighted with vector w_e for $e \in E$. To map the above formulations one can think of an entry $X_{i,j}$ equal to $x_i x_j$. Note that the above formulation can be relaxed to a semidefinite program by dropping the constraint $\operatorname{rank}(X) = 1$. It becomes then an optimization problem over an Elliptope $\mathscr{E}_n$. This relaxation was first proposed by Delorme and Poljak [6] and analyzed by Goemans and Williamson [8]. An example of feasibility sets of program 7.3 and its Delorme—Poljak relaxation can be seen in Figure 7.2.

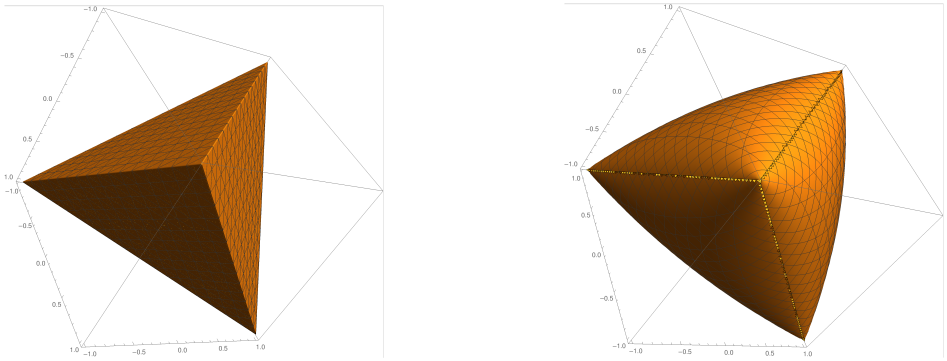


Figure 7.2: On the left, the set of feasible solution for the Max-Cut problem for $n = 3$ given by the program 7.3. On the right, the set of feasible solutions of the Delorme—Poljak relaxation of the program 7.3 for the Max-Cut problem for $n = 3$.

R The feasibility set of the program 7.3 is usually called the *cut polytope*. Its usual definition is the following:

$$\mathscr{C}_n := \operatorname{conv}\left(\{xx^\top \mid x \in \{-1,1\}^n\}\right).$$

In the following, we will study the analysis by Goemans and Williamson. Consider a feasible solution $\overline{X} \in \mathscr{S}^n$ to the constructed semidefinite program. Since the matrix $\overline{X}$ is positive semidefinite, by Proposition 9.1.4, there exists a Cholesky decomposition of $\overline{X}$ into $\overline{X} = B^\top B$. Note that one way to construct matrix B is to use eigenvalue decomposition which gives $\overline{X} = Q^\top \Lambda Q = (\Lambda^{1/2}Q)^\top(\Lambda^{1/2}Q)$, where Λ is a diagonal matrix with nonnegative entries, which are eigenvalues of $\overline{X}$, on the diagonal.

For $i \in [n]$, let $b_i \in \mathbb{R}^n$ be the i'th row of matrix $B^\top$. Note that b_i is a unite vector as $\|b_i\|_2^2 = \overline{X}_{i,i} = 1$. Throughout the section we will assume that vertices map to numbers in $[n]$ and abuse notation by writing b_v for a vertex $v \in V$. Denote by

$$\sigma_{u,v} := \langle b_u, b_v \rangle = \cos(\angle(b_u, b_v))$$

and note that for $u, v \in V$, $\sigma_{u,v} = -1$ is desirable, as it increases the objective, since $Tr(W\overline{X}) = \sum_{(u,v)\in E} w_{u,v}\frac{1-\overline{X}_{u,v}}{2} = \sum_{(u,v)\in E} w_{u,v}\frac{1-\langle b_u b_v\rangle}{2}$.

Goemans—Williamson rounding of the Delorme—Poljak semidefinite relaxation

We start with some intuition behind the Goemans Williamson rounding. We already discussed that for $u, v \in V$, $\langle b_u, b_v \rangle = -1$ is desirable. Note that $\langle b_u, b_v \rangle = -1$ for unit vectors b_u, b_v corresponds to the case $b_u = -b_v$, which is clearly not possible for all the pair of vectors b_u, b_v for $u, v \in V$. However, what is possible, and beneficial, is to have every pair b_u, b_v for $u, v \in V$ to point away from each other as much as possible.

R There is something more in the Delorme–Poljak semidefinite relaxation and the Goemans—Williamson rounding than just having vectors pointing away from each other as much as possible. The reason for that is that every edge has a weight, so we would prefer to spread out endpoints of a heavy edge at the cost of potential endpoints of some light edges getting closer. Another interesting remark is that for a complete graph K_n with weight $w_e = 1/\binom{n}{2}$ for every $e \in E$, simply spreading the vectors as much as possible is a feasible and optimal solution to the Delorme—Poljak semidefinite relaxation.

Now we are ready to discuss the Goemans—Williamson relaxation. First we sample uniformly at random a unit vector r, this can be done using e.g., Marsagila algorithm [22]. Next we take a

hyperplane H through the origin such that r is its normal vector. Finally, we construct a vector $\bar{x} \in \{-1,1\}^n$ such that for $u \in V$, $\bar{x}_u = 1$ iff $b_u \in H^+$ and $\bar{x} = -1$ iff $b_u \in H^-$. To compute the probability that a single edge $(u,v) \in E$ is cut (i.e., the hyperplane H separates vectors b_u and b_v) consider a two dimensional hyperplane containing b_u and b_v. The probability that (u,v) is in the cut is equal to the probability that a random diameter separates the vectors b_u and b_v in the two dimensional hyperplane.

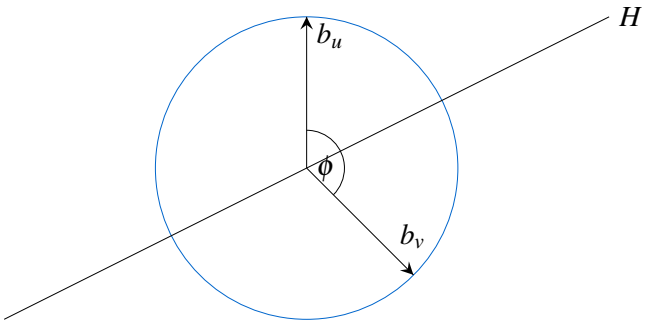


Figure 7.3: A projection on the hyperplane containing b_u and b_v.

Thus the probability that b_u and b_v are separated by H is equal to $\frac{\arccos(\langle b_u, b_v \rangle)}{\pi}$. Now, we look for α such that,

$$\mathbb{E}\left[\sum_{(u,v)\in E} w_{u,v}\left(\frac{1-\bar{x}_u\bar{x}_v}{2}\right)\right] = \sum_{(u,v)\in E} w_{u,v}\frac{\arccos(\langle b_u, b_v\rangle)}{\pi}$$
$$\geq \alpha \sum_{(u,v)\in E} w_{u,v}\left(\frac{1-\langle b_u, b_v\rangle}{2}\right)$$

and indeed, as shown in the figure below, for $\alpha = 0.87856...$ we have

$$\frac{\arccos(\sigma_{u,v})}{\pi} \geq \alpha\left(\frac{1-\sigma_{u,v}}{2}\right) \qquad \text{for every } \sigma_{u,v} \in [-1,1].$$

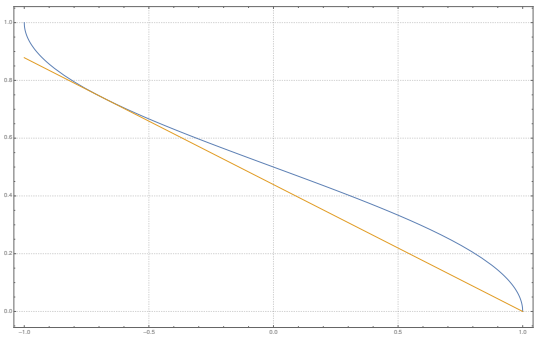


Figure 7.4: $\frac{\arccos(\sigma_{u,v})}{\pi}$ in blue and $0.87856...(1/2 - \sigma_{u,v}/2)$ in orange.

Since $\sum_{(u,v)\in E} w_{u,v}\left(\frac{1-\langle b_u, b_v\rangle}{2}\right)$ is the optimal solution of the Delorme—Poljak relaxation, which is greater equal than the size of the maximum cut in graph G, the rounded solution is an α-approximation for the Max-Cut problem.

Theorem 7.5.5 The Goemans—Williamson rounding of the Delorme—Poljak semidefinite relaxation provides a cut with an expected size at least 0.87856... times the maximum cut in the graph.

To finish this section, note that the Goemans—Williamson rounding shows that Delorme—Poljak relaxation has an integrality gap of at most $1/0.87856...$ Let us now go back to the general quadratic optimization problem over $-1/1$ variables that was presented in Equation 7.2. The following result by Nesterov bounds the integrality gap of a semidefinite relaxation of the problem presented in 7.2.

Theorem 7.5.6 For any matrix $W \in \mathbb{R}^{n \times n}$, the semidefinite relaxation of the problem presented in Equation 7.2 has integrality gap of at most $\pi/2$.

7.6 Exercises

Exercise 7.1 Prove Proposition 9.1.4.

A matrix is called *positive definite* if all its eigenvalues are positive, denoted by $A \succ 0$, i.e., $A \in \mathscr{S}^n_{++}$. Can you characterize in an analogous way positive definite matrices?

Exercise 7.2 — More difficult *. By Property 5 in Proposition 9.1.4 every positive definite matrix $A \in \mathscr{S}^n_{++}$ has a Cholesky decomposition $A = B^\top B$. Moreover, the decomposition is unique if the matrix B is upper triangular, with $B_{ii} > 0$. Show that B_{ii} is a concave function of A. Hint: B_{ii} can be represented as $B_{ii} = (w - z^\top Y^{-1} z)^{1/2}$, where

$$\begin{pmatrix} Y & z \\ z^\top & w \end{pmatrix}$$

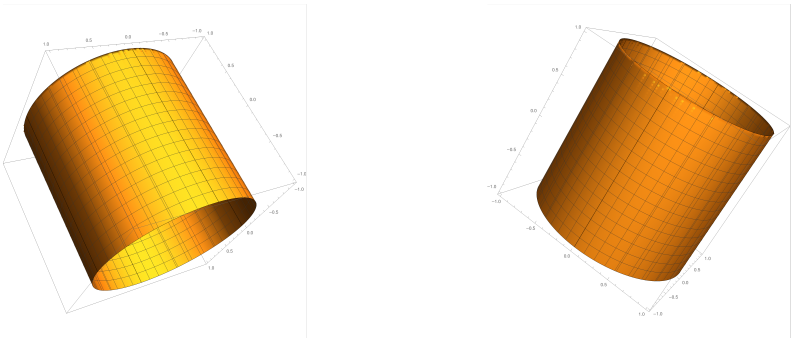
is the leading $i \times i$ submatrix of A.

Exercise 7.3 — More difficult *. A matrix $X \in \mathscr{S}^n$ is called *copositive* if $a^\top X a \geq 0$ for all $a \in \mathbb{R}^n_{\geq 0}$.

1. Show that the set of copositive matrices form a proper cone.
2. Find the dual cone of the set of copositive matrices.

Exercise 7.4 Describe the relationship between second order cone $\mathscr{Q}^2$ and the cone of positive semidefinite matrices $\mathscr{S}^2_+$.

Exercise 7.5 Consider the following set:



Is this set a spectrahedron? Can you write it as a feasibility set of a semidefinite program?

Exercise 7.6 — More difficult *. Let $\pi : \mathscr{S}^n \to \mathbb{R}^{\binom{n}{2}}$ be a projection of a matrix $X \in \mathscr{S}^n$ onto its off-diagonal entries $X_{i,j}$ for $i < j$. Elliptope $\mathscr{E}_n$ is a closed convex body, moreover it contains the origin in its interior. Thus we can define its *polar dual*, called *dual elliptope* as

$$\mathscr{E}_n^\circ := \{y \in \mathbb{R}^{\binom{n}{2}} \mid y^\top x \leq 1 \text{ for every } x \in \pi(\mathscr{E}_n)\}.$$

Prove that

$$\mathscr{E}_n^\circ = -2\pi(\mathscr{O}_n).$$

Exercise 7.7 Let $q_0(x)$ and $q_i(x)$ for $i \in [m]$ be quadratic functions in n variables. Consider the following optimization problem.

$$\begin{aligned} &\max\ q_0(x) \\ &\quad q_i(x) \leq 0 \qquad \text{for } i \in [m] \end{aligned}$$

Construct a semidefinite relaxation for this problem. Note that q_i for $i \in [m]$ do not have to be convex functions.

Exercise 7.8 — Mathematical Background. The following exercise revisits some of the mathematical background needed for this chapter, which can also be found in Appendix 9.1.

i) For the following matrices, decide whether they are positive semidefinite.

- ☐ $A_1 = 0$,
- ☐ $A_2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$,
- ☐ $A_3 = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$,
- ☐ $A_4 = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$,
- ☐ For some $v, w \in \mathbb{R}^2 \setminus \{0\}$, $\langle v, w \rangle = 0$ the matrix $A_5 = 2vv^\top - ww^\top$.

ii) Decide which of the following matrices V can be part of the eigenvalue decomposition $A = VDV^\top$ of $A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$.

- ☐ $V_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ -1 & 1 & 2 \end{pmatrix}$,
- ☐ $V_2 = \frac{1}{\sqrt{6}} \begin{pmatrix} \sqrt{3} & \sqrt{2} & 1 \\ 0 & \sqrt{2} & -2 \\ -\sqrt{3} & \sqrt{2} & 1 \end{pmatrix}$,
- ☐ $V_3 = \frac{1}{\sqrt{6}} \begin{pmatrix} \sqrt{3} & 1 & \sqrt{2} \\ 0 & -2 & \sqrt{2} \\ -\sqrt{3} & 1 & \sqrt{2} \end{pmatrix}$.

iii) Decide which of the following is a square root of $A = \begin{pmatrix} \frac{13}{4} & -\frac{5}{2} \\ -\frac{5}{2} & 2 \end{pmatrix}$?

- ☐ $B_1 = \begin{pmatrix} \frac{3}{2} & -1 \\ -1 & 1 \end{pmatrix}$,
- ☐ $B_2 = \begin{pmatrix} \frac{2}{3} & 1 \\ 1 & -1 \end{pmatrix}$,
- ☐ $B_3 = \begin{pmatrix} -1 & 1 \\ \frac{3}{2} & -1 \end{pmatrix}$,
- ☐ $B_3 = \begin{pmatrix} 1 & -1 \\ \frac{2}{3} & 1 \end{pmatrix}$.

Exercise 7.9 For the following statements decide if they are correct and give a short proof or counterexample respectively.

Let $A_0, \ldots, A_n \in \mathscr{S}^m$ and for $x \in \mathbb{R}^n$ define

$$A(x) := A_0 + x_1 A_1 + \cdots + x_n A_n \in \mathscr{S}^m.$$

We order the eigenvalue of $A(x)$ in decreasing order, i.e. $\lambda_1(x) \geq \cdots \geq \lambda_m(x)$. Decide if the following problems can be formulated as a SDP for any $A_0, \ldots, A_n$.

a) *i*) Minimize the largest eigenvalue $\lambda_1(x)$, i.e.

$$\min_{x \in \mathbb{R}^n} \lambda_1(x).$$

ii) Minimize the smallest eigenvalue $\lambda_m(x)$, i.e.

$$\min_{x\in\mathbb{R}^n} \lambda_m(x).$$

iii) Minimize the spread between the largest and smallest eigenvalue, i.e.

$$\min_{x\in\mathbb{R}^n} \lambda_1(x) - \lambda_m(x).$$

In the setting of matrix completion, one can look at the following sub-problem. Let $A \in \mathscr{S}^m$ where only a certain number of entries are specified. Then the *positive semidefinite matrix completion problem* is to determine whether the missing entries of A can be filled in such that A is PSD and if, find such entries.

b) The PSD matrix completion problem formulated above can be formulated as a SDP for any $A \in \mathscr{S}^m$, i.e. there exists a SDP which can either find a completion or determine that no completion exists.

c) *i*) The relaxed discrepancy minimization problem

$$\min_{x\in[-1,1]^n} \left\| \sum_{i=1}^n x_i A_i \right\|,$$

where the norm is the spectral norm, can be phrased as a SDP for any choice of $A_1,\ldots,A_n \in \mathscr{S}^m$.

ii) The penalized version

$$\min_{x\in[-1,1]^n} \left\| \sum_{i=1}^n x_i A_i \right\| - \frac{C}{\sqrt{n}} \|x\|_1$$

can be phrased as a SDP for $n \geq 1$ and any choice of $A_1,\ldots,A_n \in \mathscr{S}^m, C > 0$.

7.7 Supplement — algebraic properties of the PSD cone

We start this section with an example to show that every spectrahedron is a basic closed semialgebraic set.

■ **Example 7.7.1** Consider the following spectrahedron

$$S = \left\{ (x,y,z) \in \mathbb{R}^3 \;\middle|\; \begin{pmatrix} 1 & x & y & z \\ x & 1 & x & y \\ y & x & 1 & x \\ z & y & x & 1 \end{pmatrix} \succeq 0 \right\}.$$

This spectrahedron is a basic closed semialgebraic set of the following form

$$S = \left\{ (x,y,z) \in \mathbb{R}^3 \middle|\; (y+x)^2 - (z+1)(x+1) \leq 0,\ (y-x)^2 - (z-1)(x-1) \leq 0, x^2 - 1 \leq 0 \right\}$$

that is presented below.

■

Next, we prove that every spectrahedral shadow is a convex set and a semialgebraic set.

Theorem 7.7.2 A spectrahedral shadow is a convex semialgebraic set.

Proof. By Theorem 7.2.3, a spectrahedron is a basic semialgebraic set, and thus also a semialgebraic set. By Tarski—Seidenberg theorem, the projection of a semialgebraic set is a semialgebraic set as well, so every spectrahedral shadow is a semialgebraic set. By Theorem 7.2.3, every spectrahedron is a convex set. By Proposition 1.1.4, projection of a convex set is a convex set, thus every spectrahedral shadow is also a convex set. ■

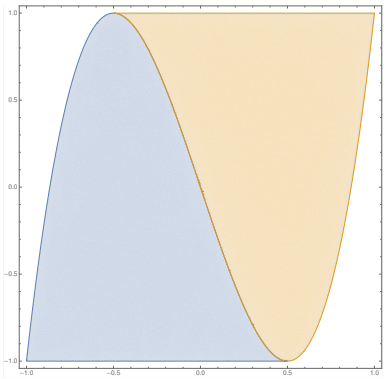
Proposition 7.7.3 The set of spectrahedral shadows is closed under projections and the convex hull of unions.

Recall that by the Tarski—Seidenberg theorem (Theorem 2.6.6) projection of every semialgebraic set is a semialgebraic set. This, however, as shown in the example below, is not true for basic semialgebraic sets, in general.

■ **Example 7.7.4** Consider a spectrahedron S as in Example 7.7.1 which is a basic semialgebraic set, as presented below on the left. However, its projection on the (x,z) plane is not a basic closed semialgebraic set. It is the union of basic closed semialgebraic sets of the following form

$$S_{x,z} = \left\{ (x,z) \in \mathbb{R}^2 \middle|\; -1 \leq z \leq 4x^3 - 3x,\ x \leq \frac{1}{2} \right\} \cup \left\{ (x,z) \in \mathbb{R}^2 \middle|\; 4x^3 - 3x \leq z \leq 1,\ -\frac{1}{2} \leq x \right\}$$

that can be seen below on the right.



■

The connection between convex semialgebraic sets and spectrahedral shadows has been a subject of intense study. The following conjecture, that can be seen as a partial converse of Theorem 7.7.2, has been widely believed.

Conjecture 7.7.5 — Helton—Nie 2009 (12). For every $n \in \mathbb{N}$, every convex semialgebraic set in $\mathbb{R}^n$ is a spectrahedral shadow.

This conjecture, if true, would suggest that semidefinite programming is a universal method to approach convex polynomial optimization problems. However, this conjecture was recently disproved by Schreiderer.

Theorem 7.7.6 — Schreiderer 2018 (17). For any $n > 2$, there exists a convex semialgebraic set in $\mathbb{R}^n$, which is not a spectrahedral shadow.

7.8 Solutions

Solution of the Exercise 7.1

In the following we do not prove Proposition 9.1.4. We only characterize positive definite matrices.

The following statements are equivalent:

1. The matrix $A \in \mathscr{S}^n_{++}$ is *positive definite*, denoted by $A \prec 0$, i.e., $A \in \mathscr{S}^n_{++}$.
2. For all $x \in \mathbb{R}^n$, $x^\top A x > 0$.
3. All eigenvalues of A are positive.
4. All n leading principal minors of A are positive.
5. There exists a factorization $A = B^\top B$, with B square and nonsingular.

Solution to the Exercise 7.2

We use the following hints to solve the exercise.

Hint 1: B_{ii} can be represented as $B_{ii} = (w - z^T Y^{-1} z)^{1/2}$, where

$$\begin{bmatrix} Y & z \\ z^T & w \end{bmatrix}$$

is the leading $i \times i$ submatrix of A.

Hint 2: A function $f : X \to \mathbb{R}$ is convex if and only if its *epigraph*

$$\operatorname{epi}(f) := \{ (x,t) \in X \times \mathbb{R} : \ t \geq f(x) \}$$

is a convex set.

We want to show that the function (recall that $\mathscr{S}^n_{++}$ is the set of $n \times n$ positive definite matrices, see solution of Exercise 7.1)

$$f : \begin{cases} \mathbb{R}^n \times \mathscr{S}^n_{++} \to \mathbb{R}, \\ (z, Y) \mapsto z^T Y^{-1} z, \end{cases}$$

where $\mathscr{S}^n_{++}$ is the cone of PD matrices, is a convex function. For any matrix $Y \in \mathscr{S}^n_{++}$, the Schur complement condition implies

$$\begin{bmatrix} Y & z \\ z^T & w \end{bmatrix} \succeq 0 \quad \Leftrightarrow \quad w - z^T Y^{-1} z \geq 0.$$

This allows us to formulate the epigraph of f as

$$\begin{aligned} \operatorname{epi}(f) &= \{ (z, Y, w) : \ Y \succ 0, \ w \geq z^T Y^{-1} z \} \\ &= \left\{ (z, Y, w) : \ Y \succ 0, \ \begin{bmatrix} Y & z \\ z^T & w \end{bmatrix} \succeq 0 \right\}. \end{aligned}$$

This set is defined by LMIs and is thus a convex set, meaning that f is a convex function and $-f$ a concave function. The identity function is clearly a concave function as well, meaning that the function

$$(z, Y, w) \mapsto w - z^T Y^{-1} z$$

is concave as a sum of two concave function. Lastly, because the function $x \mapsto x^{1/2}$ is increasing and concave, the function

$$(z, Y, w) \mapsto (w - z^T Y^{-1} z)^{1/2} =: B_{ii}$$

is concave by Exercise 1.3.

Solution to the Exercise 7.3

Hint: A matrix $A \in \mathscr{S}^n$ is *completely positive (CP)* if it is of the form $A = \sum_{i=1}^{s} l_i l_i^T$ for some $s \in \mathbb{N}$ and $l_1, \ldots, l_s \in \mathbb{R}^n_{\geq 0}$.

Part 1.

Let $\mathscr{C}^n$ be the set of all copositive matrices in $\mathscr{S}^n$. We investigate all properties of proper cones.

The set $\mathscr{C}^n$ is clearly a convex cone. Indeed, for $X, Y \in \mathscr{C}^n$, $\lambda \geq 0$ and any vector $a \in \mathbb{R}^n_{\geq 0}$,

$$a^T(\lambda X + Y)a = \lambda a^T X a + a^T Y a \geq 0.$$

It is also obvious that $\mathscr{C}^n$ is solid since any PSD matrix is copositive, and $\mathscr{S}^n_+$ is a proper (and in particular solid) cone.

Lastly, to show that $\mathscr{C}^n$ is pointed, we prove $\mathscr{C}^n \cap -\mathscr{C}^n = \{0\}$. For this, let $X \in \mathscr{C}^n \cap -\mathscr{C}^n$ which implies $a^T X a = 0$ for every $a \in \mathbb{R}^n_{\geq 0}$. We also define $e^{(i,j)} \in \mathbb{R}^n$ by $e^{(i,j)}_k = 1$ if $k = i$ or $k = j$ and $e^{(i,j)}_k = 0$ otherwise. Since $e^{(i,j)} \in \mathbb{R}^n_{\geq 0}$ for every $i, j \in [n]$, it follows that $X_{ii} = (e^{(i,i)})^T X e^{(i,i)} = 0$ and $2X_{ij} = (e^{(i,j)})^T X e^{(i,j)} = 0$ for $i \neq j$, meaning that $X = 0$.

Part 2

Let $\mathscr{C}^n$ be the set of copositive matrices and $\mathscr{P}^n$ be the set of CP matrices.

We show that $(\mathscr{P}^n)^* = \mathscr{C}^n$. Let $A \in \mathscr{P}^n$ and $B \in (\mathscr{P}^n)^*$. Without loss of generality, we can assume that $A = ll^T$ with $l \in \mathbb{R}^n_{\geq 0}$, so we have

$$0 \leq \langle A, B \rangle = \sum_{i,j=1}^{n} A_{ij} B_{ij} = \sum_{i,j=1}^{n} B_{ij} l_i l_j = l^T B l.$$

Since l can be chosen arbitrarily in $\mathbb{R}^n_{\geq 0}$, the only condition set to the matrix $B \in \mathscr{S}^n$ is to be copositive, which confirms $(\mathscr{P}^n)^* = \mathscr{C}^n$.

Now, it is simple to show that $\mathscr{P}^n$ is a convex cone, which implies

$$(\mathscr{C}^n)^* = (\mathscr{P}^n)^{**} = \overline{\mathscr{P}^n}$$

by Proposition 1.3.2. Actually, $\mathscr{P}^n$ is a closed cone, as shown in [10, Lemma 16.2.1], so $(\mathscr{C}^n)^* = \mathscr{P}^n$, but the proof is too extensive to be covered in this exercise.

Solution to the Exercise 7.4

First let us describe the cone $\mathscr{S}^2_+$. For this, we interpret $\mathscr{S}^2$ as a subspace of $\mathbb{R}^3$ by representing each matrix

$$A = \begin{pmatrix} y & x \\ x & z \end{pmatrix} \in \mathscr{S}^2$$

as a vector $(x, y, z) \in \mathbb{R}^3$. By Sylvester's criterion, A is PSD if and only if $\det(A) = yz - x^2 \geq 0$ and $x, y \geq 0$, meaning that the the cone $\mathscr{S}^2_+$ is the subset of $\mathbb{R}^3$ given as

$$\tilde{\mathscr{S}}^2_+ := \left\{ (x, y, z) \in \mathbb{R} \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} : \; yz \geq x^2 \right\}.$$

Note that this set looks almost identical to the rotated second-order cone

$$\mathscr{Q}^n_{rot} := \left\{ (x, y, z) \in \mathbb{R}^n \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} : \; 2yz \geq \|x\|_2^2 \right\}$$

for $n = 1$. With the substitution $x' = \sqrt{2}x$, we have $(x, y, z) \in \tilde{\mathscr{S}}^2_+$ if and only if $(x', y, z) \in \mathscr{Q}^1_{rot}$. By Theorem 4.1.1, it follows that

$$(x, y, z) \in \tilde{\mathscr{S}}^2_+ \quad \text{if and only if} \quad (2x, y - z, y + z) \in \mathscr{Q}^2.$$

Therefore the two-dimensional second-order cone is the image of $\tilde{\mathscr{S}}^2_+$ through a linear automorphism, meaning that the sets $\mathscr{S}^2_+$ and $\mathscr{Q}^2$ are linearly isomorphic.

Solution to the Exercise 7.5

A cylinder is a spectrahedron, as it can be described by the following system of inequalities:

$$C =: \{(x,y,z) \in \mathbb{R}^3 : \ x^2+y^2 \leq 1, \ -1 \leq z \leq 1\}$$

which corresponds to the intersection of the cone of semidefinite matrices $\mathscr{S}^4$ and an affine subspace defined in the following way:

$$C := \{(x,y,z) \in \mathbb{R}^3 : \ \begin{bmatrix} 1-x & y & 0 & 0 \\ y & x+1 & 0 & 0 \\ 0 & 0 & 1-z & 0 \\ 0 & 0 & 0 & 1+z \end{bmatrix} \succeq 0\}.$$

Solution to the Exercise 7.6

Let us define the following sets:

$$\begin{aligned} \mathscr{E}_n &= \{X \in \mathscr{S}^n_+ \mid X_{i,i} = 1 \quad \forall\, i \in [n]\} \\ \mathscr{O}_n &= \{X \in \mathscr{S}^n_+ \mid \mathrm{Tr}(X) = 1\} \\ \mathscr{E}^*_n &= \{Y \in \mathscr{S}^n \mid \langle Y, X\rangle \leq 1 \quad \forall\, X \in \mathscr{E}_n\} \\ \mathscr{E}^\circ_n &= \{y \in \mathbb{R}^{\binom{n}{2}} \mid \langle y, x\rangle \leq 1 \quad \forall\, x \in \pi(\mathscr{E}_n)\} \end{aligned}$$

First of all notice that for $X, Y \in \mathscr{S}^n_+$, the following holds.

$$\begin{aligned} \langle Y, X\rangle = \mathrm{Tr}(YX) &= \sum_{i=1}^n Y_{i,i}X_{i,i} + 2 \cdot \sum_{i<j} Y_{i,j}X_{i,j} \\ &= \sum_{i=1}^n Y_{i,i}X_{i,i} + 2 \cdot \pi(Y)^T \pi(X) \\ &= \sum_{i=1}^n Y_{i,i}X_{i,i} + 2 \cdot \langle \pi(Y), \pi(X)\rangle \end{aligned}$$

This means that we can find a relation between $\mathscr{E}^*_n$ and $\mathscr{E}^\circ_n$. We divide the proof into three steps.

- **Step 1.** $\mathscr{O}_n = \mathrm{conv}(bb^t \mid b \text{ unit vector})$.
  If $X \in \mathscr{O}_n$, then we can write $X = \sum_i \lambda_i v_i v_i^T$, where the λ_i for $i \in [n]$ are the eigenvalues of X and v_i are the corresponding unit eigenvectors. Since $\mathrm{Tr}(X) = 1$ then $\sum_i \lambda_i = 1$ and $\lambda_i \geq 0$ because $X \in \mathscr{S}^n_+$. Then X can be viewed as the convex combination of $v_i v_i^T$, where v_i for $i \in [n]$ are unit vectors. Hence $X \in \mathrm{conv}(bb^t \mid b \text{ unit vector})$.
  Conversely, let $X = \sum_i \lambda_i v_i v_i^T$ be a convex combination in the set $\mathrm{conv}(bb^t \mid b \text{ unit vector})$, with $\lambda_i \geq 0$ for all $i \in [n]$ and $\sum_i \lambda_i = 1$. Then $X \in \mathscr{S}^n$ because each term $v_i v_i^T$ is symmetric. Moreover since $\lambda_i \geq 0$ for all $i \in [n]$ then $X \in \mathscr{S}^n_+$. From $\sum_i \lambda_i = 1$ we get that $\mathrm{Tr}(X) = 1$.
- **Step 2.** $\mathscr{E}^*_n = \{D - M \mid M \succeq 0,\ D \text{ diagonal},\ \mathrm{Tr}(D) = 1\}$.
  It easy to prove that the set on the right-hand-side is contained in $\mathscr{E}^*_n$. Indeed

$$\begin{aligned} \langle (D-M), X\rangle &= \mathrm{Tr}((D-M)X) = \mathrm{Tr}(DX) - \mathrm{Tr}(MX) \\ &= \sum_i D_{i,i}X_{i,i} - \mathrm{Tr}(MX) = 1 - \mathrm{Tr}(MX) \leq 1 \end{aligned}$$

  for each $X \in \mathscr{E}_n$.
  Conversely, let $Y \in \mathscr{E}^*_n$. Suppose by contradiction that for each diagonal matrix D with $\mathrm{Tr}(D) = 1$, the matrix $D - Y$ is not PSD. Then it must have at least one eigenvalue less

than zero, hence $\lambda_{\min}(D-Y) < 0$. Let D_0 the diagonal matrix D such that $\lambda_{\min}(D-Y)$ is maximum, and let $\lambda_0 = \lambda_{\min}(D_0 - Y) < 0$. Let $v_1, \dots, v_k$ the eigenvectors of λ_0 such that the diagonal of $X = \sum_h v_h v_h^T$ has only ones. Therefore $X \in \mathscr{E}_n$.
Now we can get to a contradiction by getting $\langle Y, X\rangle > 1$. Indeed

$$\begin{aligned}\langle Y - D_0, X\rangle &= \mathrm{Tr}((Y-D_0)X) = \mathrm{Tr}\left((Y-D_0)\sum_h v_h v_h^T\right)\\ &= \sum_h \mathrm{Tr}((Y-D_0)v_h v_h^T) = -\sum_h \lambda_h \mathrm{Tr}(v_h v_h^T) = -n\lambda_h.\end{aligned}$$

Then

$$\begin{aligned}\langle Y, X\rangle &= \langle D_0, X\rangle + \langle Y - D_0, X\rangle\\ &= 1 + (-n\lambda_0 = 1 - n\lambda_0 > 1,\end{aligned}$$

since $\lambda_0 < 0$. This is a contradiction, then there exists a diagonal matrix D with $\mathrm{Tr}(D) = 1$ such that $D - Y$ is PSD.

- **Step 3.** $y \in \mathscr{E}_n^\circ \iff y \in \mathrm{conv}(-2\pi(bb^T) \mid b \text{ is a unit vector})$.
  First of all, given $y \in \mathbb{R}^{\binom{n}{2}}$ let us construct a symmetric matrix Y such that $\pi(Y) = y$ and $Y_{i,i} = -\frac{1}{n}$ for all $i \in [n]$.
  Then, our claim is that $y \in \mathscr{E}_n^\circ \iff Y \in \mathscr{E}_n^*$. Indeed

$$\begin{aligned}\langle X, Y\rangle = \mathrm{Tr}(XY) &= \sum_i X_{i,i}Y_{i,i} + 2\cdot\langle x, y\rangle \qquad \forall X \in \mathscr{E}_n,\ x = \pi(X)\\ &= -1 + 2\cdot\langle x, y\rangle\end{aligned}$$

  and therefore

$$\langle X, Y\rangle \le 1 \qquad \iff \qquad \langle x, y\rangle \le 1$$

  Moreover, by Step 2, $Y \in \mathscr{E}_n^* \iff Y = D - \sum_h \lambda_h b_h b_h^T$, where D is a diagonal matrix, $\mathrm{Tr}(D) = 1$, b_h for $h \in [n]$ are unit vectors and $\lambda_h \ge 0$ for all $h \in [n]$. Taking the traces of this expression we get $\mathrm{Tr}(Y) = \mathrm{Tr}(D) - \mathrm{Tr}\left(\sum_h \lambda_h b_h b_h^T\right)$, that is $-1 = 1 + \sum_h \lambda_h$. Then $\sum_h \lambda_h = 2$.
  Since $y \in \mathscr{E}_n^\circ \iff Y \in \mathscr{E}_n^*$, we can write this last fact in terms of vectors in $\mathbb{R}^{\binom{n}{2}}$:

$$y \in \mathscr{E}_n^\circ \iff y = -\sum_h \lambda_h \pi(b_h b_h^T)$$

  where $\sum_h \lambda_h = 2$. Therefore:

$$y \in \mathscr{E}_n^\circ \iff y \in \mathrm{conv}(-2\pi(bb^T) \mid b \text{ is a unit vector}).$$

This concludes the proof, since $\mathrm{conv}(-2\pi(bb^T) \mid b \text{ is a unit vector}) = -2\mathscr{O}_n$.

Solution to the Exercise 7.7

Since the functions q_i are quadratic for $i \in \{0, 1 \dots, m\}$, they can be written as $q_i(x) = x^T Q_i x + 2p_i^T x + r_i$ for $Q_i \in \mathscr{S}^n$ (note that Q_i is not necessarily PSD), $p_i \in \mathbb{R}^n$ and $r_i \in \mathbb{R}$ for all i.

For each $i \in \{0, 1 \dots, m\}$, we define the affine function $L_i \colon \mathscr{S}^n \times \mathbb{R}^n$ as

$$L_i(X, x) \coloneqq \langle X, Q_i\rangle + 2p_i^T x + r_i.$$

Since $x^T Q_i x = \mathrm{Tr}(xx^T Q_i) = \langle xx^T, Q_i \rangle$, we have $L_i(xx^T, x) = q_i(x)$ for every $x \in \mathbb{R}^n$ and $i \in \{0, 1 \ldots, m\}$, meaning that our initial problem can be reformulated as

$$\begin{aligned} &\max L_0(X,x) \\ &\quad L_i(X,x) \leq 0 \quad \text{for } i \in [m]. \\ &\quad X = xx^T. \end{aligned}$$

Since the maps L_i are affine, we can relax this problem by removing the constraint $X = xx^T$ but still require $X \succeq 0$. This results in the problem

$$\begin{aligned} &\max L_0(X,x) \\ &\quad L_i(X,x) \leq 0 \quad \text{for } i \in [m]. \\ &\quad X \succeq 0, \end{aligned}$$

which is an SDP.

Solution to the Exercise 7.8

i) A_1 True: $\sigma(A_1) = \{0, 0\}$
A_2 True: $\sigma(A_2) = \{0, 2\}$
A_3 False: $\sigma(A_4) = \{0, -2\}$
A_4 True: $\sigma(A_3) = \{0, 2\}$
A_5 False: $\sigma(A_5) = \left\{ \frac{2}{\|v\|}, -\frac{1}{\|w\|} \right\}$

ii) V_1 False: First and last column are not orthogonal.
V_2 True: One can calculate $V^\top AV = \mathrm{diag}(1, 4, 1)$
V_3 True: V_3 can be created by swapping columns of V_2

iii) This Exercise can easily be solved by calculating B_i^2 and checking whether it matches A. Alternatively a decomposition can be calculated, which will however not be trivial for this matrix.

Solution to the Exercise 7.9

a) *i)* *True.* Using that $\lambda_1(x) \leq t$ is equivalent to $A(x) \preceq tI_m$ and get the SDP

$$\begin{aligned} &\min t \\ &s.t.\ A(x) \preceq tI_m \end{aligned}$$

ii) *False.* The smallest eigenvalue is in general concave function. Hence, if we can find A_i's for which the function is not linear we are done. There are many possibilities to do so, one example could be

$$n = 2,\ A_0 = 0,\ A_1 = \begin{pmatrix} 1 & 3 \\ 3 & 2 \end{pmatrix},\ A_2 = \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}.$$

In this case we get for

$$x = \begin{pmatrix} -1 \\ -1 \end{pmatrix},\ y = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

that $\lambda_m(x) = -8,\ \lambda_m(0) = \lambda_m(y) = 0$ and hence

$$\lambda_m\left(\frac{1}{2}x + \left(1 - \frac{1}{2}\right)y\right) = 0 > -4 = \frac{1}{2}\lambda_m(x) + \left(1 - \frac{1}{2}\right)\lambda_m(y).$$

iii) *True.* With the same argument as in a) *i*) we get the SDP

$$\begin{aligned} &\min t_1 - t_2 \\ &s.t.\ t_2 I_m \preceq A(x) \preceq t_1 I_m \end{aligned}$$

b) *True.* The goal can be formulated as the LMI

$$A + \sum_{(i,j)\in U} a_{i,j} E_{i,j} \succeq 0,$$

where U is the set of unspecified indices and $(E_{i,j})_{k,l} = (E_{j,i})_{k,l} = \mathbb{1}_{\{\{i,j\}\}}(\{k,l\})$. Hence we can use the SDP

$$\begin{aligned} &\min 1 \\ &s.t.\ A + \sum_{(i,j)\in U} a_{i,j} E_{i,j} \succeq 0. \end{aligned}$$

c) *i*) *True.* We can first rewrite the problem to

$$\begin{aligned} &\min_{\substack{x\in[-1,1]^n \\ s\in\mathbb{R}}} s \\ &\quad s.t.\ \|A(x)\| \leq s, \end{aligned}$$

where $A(x) \coloneqq \sum_{i=1}^n x_i A_i$. Since the maximum eigenvalue function is convex, and the minimum eigenvalue function concave we can write this as the SDP

$$\begin{aligned} &\min s \\ &s.t.\ -sI_m \preceq H(x) \preceq sI_m \\ &\quad -1 \leq x_i \leq 1 \qquad \text{for } i \in [n]. \end{aligned}$$

ii) *False.* Choosing $n = 1, C = 1, A_0 = A_1 = 0$ gives us the objective

$$-\frac{1}{\sqrt{n}}|x|$$

which is nonconvex.

8. Geometric programs

In this chapter, we study another class of programs that can be cast as convex optimization programs, namely, geometric programs. Geometric programs are not convex in their natural form; however, they can be transformed into convex form. In this chapter we study the log-log transformation that can be used to convexify geometric programs.

We start with basic definitions.

Definition 8.0.1 The function $f: \mathbb{R}^n_{\geq 0} \to \mathbb{R}$ defined as

$$f(x) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}$$

where $c \in \mathbb{R}_{\geq 0}$ and $a \in \mathbb{R}^n$ is called a *monomial function*.

Note that the monomial function should not be confused with a well-known definition of a monomial in which the vector a can have only nonnegative integer entries. The set of monomial functions is closed under multiplication and division, see Exercise 8.1. A summation of monomial functions forms a structure called posynomial function.

Definition 8.0.2 The function $f: \mathbb{R}^n_{\geq 0} \to \mathbb{R}$ defined as

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{k1}} x_2^{a_{k2}} \cdots x_n^{a_{kn}}$$

where $c_k \in \mathbb{R}_{\geq 0}$ and $a_k \in \mathbb{R}^n$ for all $k \in [K]$ is called a *posynomial function*.

Every monomial function in the summation in a posynomial function is called a term. The set of posynomial functions is closed under addition and multiplication, see Exercise 8.1. We are ready to present a general definition of a geometric program (GP).

Definition 8.0.3 Let f, g_i for all $i \in [m]$ be posynomial functions and let h_j for all $j \in [\ell]$ be

monomial functions. The *Geometric Program (GP)* in general form is the following program:

$$\begin{aligned} \min\ & f(x) \\ & g_i(x) \le 1, \qquad \text{for } i \in [m], \\ & h_j(x) = 1, \qquad \text{for } j \in [\ell], \\ & x \in \mathbb{R}^n_{\ge 0}. \end{aligned}$$

A simple consequence of Exercise 8.1 is that posynomial function divided by a monomial function is a posynomial function and monomial function divided by a monomial function is a monomial function. Thus even more general programs can be transformed into geometric programs. As an example for f_i's being posynomial functions and h_i, h_j, h_k's being monomial functions, the following program

$$\begin{aligned} \min\ & f(x) \\ & g_i(x) \le h_i(x), \qquad \text{for } i \in [m], \\ & h_j(x) = h_k(x), \qquad \text{for } j,k \in [\ell], \\ & x \in \mathbb{R}^n_{\ge 0}. \end{aligned}$$

is also a geometric program. Indeed, by Exercise 8.1, the constraint $f_i(x) \le h_i(x)$ can be replaced by $f_i(x)/h_i(x) \le 1$ which is a posynomial inequality and the constraint $h_j(x) = 1$ can be replaced by $h_j(x)/h_k(x) = 1$ which is a monomial equality.

■ **Example 8.0.4** Examples of Monomials are $xy, 3\sqrt{xy}z^2$ or $1.5 \cdot \frac{xy^{\frac{4}{3}}}{z^2}$ but $x+y$ or $-x$ are none. Consequently $xy + 3\sqrt{xy}z^2 + 1.5\frac{xy^{\frac{4}{3}}}{z^2}$ and $x+y+z$ are posynomials but $x-y$ is not. ■

■ **Example 8.0.5** An example of a geometric program in standard form is

$$\begin{aligned} \min\ & x\sqrt{y} \\ & xyz = 1 \\ & xy + yz \le 1 \\ & x + \frac{z}{y} \le 1. \end{aligned}$$

Next the program

$$\begin{aligned} \max\ & yz \\ & 2 \le x \le 3 \\ & x + y \le xz \\ & \sqrt{yz} = x \end{aligned}$$

is not a geometric program by the definition above, but it is easily seen to be equivalent to the geometric program:

$$\begin{aligned} \max\ & yz \\ & \frac{x}{3} \le 1 \\ & \frac{2}{x} \le 1 \\ & \frac{1}{z} + \frac{y}{xz} \le 1 \\ & \frac{\sqrt{yz}}{x} = 1. \end{aligned}$$

Finally consider the similar

$$\begin{aligned}\max\ & yz+xz\\ & 2\le x+y\le 3\\ & x+y\le \sqrt{x}+z\\ & \sqrt{yz}=x^2+y^2.\end{aligned}$$

then the analogous reductions as above do **not** give a geometric program. For example for the first constraint we can transform $x+y\le 3$ to $\frac{x}{3}+\frac{y}{3}\le 1$ but transforming $2\le x+y$ is not possible. One could try $\frac{2}{x+y}\le 1$ but posynomials may only contain monomials in the denominator. On the other hand $-\frac{x}{2}-\frac{y}{2}\le 1$ doesn't work either as posynomial may only use positive coefficients. (The pos in posynomial stands for positive.) ∎

It is important to note that according to Definition 2.1.2 of convex program, geometric program in general form, as presented in Definition 8.0.3, is not a convex program. In the following, we present an explicit example.

∎ **Example 8.0.6** Consider a posynomial function $f:\mathbb{R}\to\mathbb{R}$ defined as

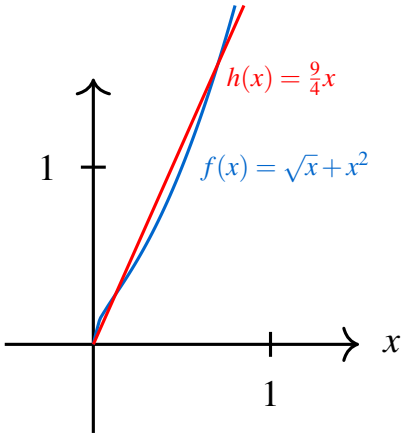
$$f(x):=\sqrt{x}+x^2.$$

It is easy to check that $f(0)=0$ and $f(1/4)=1/2+1/16=9/16$. To show that the function f is not convex it is enough to note that

$$\frac{3}{4}\cdot f(0)+\frac{1}{4}\cdot f\left(\frac{1}{4}\right)=\frac{9}{64}\le f\left(\frac{3}{4}\cdot 0+\frac{1}{4}\cdot\frac{1}{4}\right)=\frac{1}{4}+\frac{1}{256}=\frac{65}{256}.$$

Since function f is not convex, a geometric program optimizing a posynomial f over some feasibility set described by inequalities involving posynomial functions is not a convex program.

Moreover, as easy to check, the function f is also not a concave function. To visualize the statements see the plot below.



∎

As mentioned above, a geometric program, as defined in Definition 8.0.3 is not a convex program. However, it is possible to transform the program in Definition 8.0.3 into a convex program. From a broader perspective, transforming an optimization program into an explicitly described convex program with inequalities involving convex functions is not an easy task. An example of such transformation was discussed in Chapter 2 in Example 2.1.5, Example 2.1.6 and in Exercise 2.2. All the mentioned transformations apply to specific examples, as there is no one-size-fits-all technique to transform the program into convex form. Nevertheless, there exist techniques that can be used successfully in a wide range of situations. The first step towards building such a technique is understanding the concept of a log transformation.

8.1 Log-convex functions

We directly start with the definition of a logarithmically convex function.

Definition 8.1.1 Let $X \subseteq \mathbb{R}^n$ be a convex set. A function $f : X \to \mathbb{R}$ is *logarithmically convex (log-convex)* if $f(x) > 0$ for all $x \in X$ and $\log f$ is convex.

Moreover, if a function f is such that $1/f$ is log-convex, we call the function *logarithmically concave (log-concave)*.

It is possible to verify log-convexity without directly referring to convexity and logarithmic transformation.

Theorem 8.1.2 Let $X \subseteq \mathbb{R}^n$ be a convex set. A function $f : X \to \mathbb{R}$, such that $f(x) > 0$ for all $x \in X$, is log-convex if and only if for all $x, y \in X$ and $\lambda \in [0,1]$ it holds that

$$f(\lambda x + (1-\lambda)y) \leq f(x)^{\lambda} \cdot f(y)^{1-\lambda}.$$

Proof. By definition f is log-convex if and only if $\log f$ is convex. The latter is by definition equivalent to

$$\log f(\lambda x + (1-\lambda)y) \leq \lambda \log f(x) + (1-\lambda) f(y)$$

for all $x, y \in X$ and $\lambda \in [0,1]$. Taking exponentials on both sides, using that f is positive, we see that this equation is equivalent to

$$f(\lambda x + (1-\lambda)y) = e^{\log f(\lambda x + (1-\lambda)y)} \leq e^{\lambda \log f(x) + (1-\lambda)\log f(y)} = f(x)^{\lambda} \cdot f(y)^{1-\lambda}$$

finishing the proof. ■

Log-convex (log-concave) functions have many interesting properties. Some of them are listed below.

Proposition 8.1.3 Let f_1, f_2 and g_1, g_2 be log-convex and log-concave functions, respectively. Let f and g be a convex and concave function, respectively. The following properties hold:

1. g is a log-concave function, f is not necessarily a log-convex function,
2. f_1 is a convex function, g_1 is not necessarily a concave function,
3. $f_1 \cdot f_2$ is a log-convex function, $g_1 \cdot g_2$ is a log-concave function,
4. $a \cdot f_1$ is a log-convex and $a \cdot g_1$ is a log-concave function for all $a \in \mathbb{R}_{\geq 0}$,
5. $f_1 + f_2$ is a log-convex function, $g_1 + g_2$ is not necessarily a log-concave function.

Proof.

1. The first part follows directly by applying the composition rule from Exercise 1.3 for a concave non-decreasing logarithm function with a concave function g. For the second part consider a function $f(x) = x$ that is convex however it is not log-convex as $\log f(x) = \log x$ which is a concave function.
2. The first part follows by the second task in Exercise 1.3 since $f(x) = \exp(\log f(x))$ and $\exp(x)$ is a convex non-decreasing function and $log(x)$ is a concave function. The second part follows by letting $g_1(x) = x^2$ that is a log-concave function, as $\log(x^2) = 2\log x$, but not a concave function.
3. By Definition 8.1.1 we have that

$$\log(f_1(\lambda x + (1-\lambda)y)) \leq \lambda \log(f_1(x)) + (1-\lambda)\log(f_1(y)).$$

and

$$\log(f_2(\lambda x + (1-\lambda)y)) \leq \lambda \log(f_2(x)) + (1-\lambda)\log(f_2(y)).$$

Summing up both inequalities, we get

$$\log\left(f_1 \cdot f_2\left(\lambda x+(1-\lambda)y\right)\right) \leq \lambda \log\left(f_1 \cdot f_2\left(x\right)\right)+(1-\lambda)\log\left(f_1 \cdot f_2\left(y\right)\right).$$

Taking the *exp* transformation on both sides finishes the proof for log-convex functions

$$f_1 \cdot f_2\left(\lambda x+(1-\lambda)y\right) \leq f_1 \cdot f_2\left(x\right)^{\lambda} \cdot f_1 \cdot f_2\left(y\right)^{1-\lambda}.$$

The proof for log-concave functions is analogous.

4. Follows directly from the Definition 8.1.1.
5. For the first part we start by applying Theorem 8.1.2 which yields

$$\left(f_1+f_2\right)\left(\lambda x+(1-\lambda)y\right) \leq f_1(x)^{\lambda} f_1(y)^{1-\lambda}+f_2(x)^{\lambda} f_2(y)^{1-\lambda}.$$

Thus the statement follows if we can prove that for all $a,b,c,d>0$ and $\lambda \in [0,1]$

$$a^{\lambda} b^{1-\lambda}+c^{\lambda} d^{1-\lambda} \leq (a+c)^{\lambda} \cdot (b+d)^{1-\lambda} \tag{8.1}$$

holds. Therefore we will need Young's inequality which shows that

$$x^{\lambda} y^{1-\lambda} \leq \lambda x+(1-\lambda)y. \tag{8.2}$$

First we divide (8.1) by $(a+c)^{\lambda}$ and $(b+d)^{1-\lambda}$ to get

$$\left(\frac{a}{a+c}\right)^{\lambda}\left(\frac{b}{b+d}\right)^{1-\lambda}+\left(\frac{c}{a+c}\right)^{\lambda}\left(\frac{d}{b+d}\right)^{1-\lambda} \leq 1.$$

This reduces the problem to $\tilde{a},\tilde{b},\tilde{c},\tilde{d}>0$ with $\tilde{a}+\tilde{c}=\tilde{b}+\tilde{d}=1$, i.e. we need to show

$$\tilde{a}^{\lambda}\tilde{b}^{1-\lambda}+\tilde{c}^{\lambda}\tilde{d}^{1-\lambda} \leq 1.$$

Applying (8.2) to both terms on the left yields

$$\begin{aligned}\tilde{a}^{\lambda}\tilde{b}^{1-\lambda}+\tilde{c}^{\lambda}\tilde{d}^{1-\lambda} &\leq \lambda\tilde{a}+(1-\lambda)\tilde{b}+\lambda\tilde{c}+(1-\lambda)\tilde{d}\\ &= \lambda\left(\tilde{a}+\tilde{c}\right)+(1-\lambda)\left(\tilde{b}+\tilde{d}\right)=1,\end{aligned}$$

finishing the proof of the first part.
For the second part consider two exponential random variables defined as $g_1(x)=\exp(-x)$ and $g_2(x)=2\exp(-2x)$. Both functions are clearly log-concave, however the summation of g_1+g_2 is not log-concave. In fact one can show that the summation of two exponential random variables is a log-convex function.

■

Here we give a list of examples of log-convex and log-concave functions.

■ **Example 8.1.4** List of log-convex and log-concave functions.

- *Affine function.* $f:\mathbb{R}^n \to \mathbb{R}$ defined as $f(x)=a^{\top}x+b$ is log-concave on $\{x|a^{\top}x+b>0\}$.
- *Powers.* $f:\mathbb{R}_{\geq 0} \to \mathbb{R}$ defined as $f(x)=x^a$ is log-convex for $a \leq 0$ and log-concave for $a \geq 0$.
- *Exponentials.* $f:\mathbb{R} \to \mathbb{R}$ defined as $f(x)=e^{ax}$ is log-convex and log-concave.
- *Determinant.* $f:\mathscr{S}^n_{++} \to \mathbb{R}$ defined as $f(X)=det(X)$ is log-concave.

■

It is worth mentioning that the log-transformation can be applied not only to the given function f but also to $\log f$. In polynomial optimization, it is called a log-log-transformation. Moreover, it is possible to apply the log-transformation iteratively; however, as mentioned in Proposition 8.1.3, a function that is log-convex is also convex. Thus, it is impossible to convexify a non-convex function by directly using the log-transformation. In the next section, we study another generalization of the log-transformation that will be critical in transforming geometric programs into convex programs. Unfortunately, it is also called a log-log transformation. To avoid confusion, from this point on, whenever we use the name log-log transformation, we have in mind the concept discussed in the following section.

8.2 Log-log transformation

In this subsection we study the log-log transformation of functions. The log-log transformation of a function $f(x)$ consist in changing the variables x_i into $y_i = log\ x_i$ (thus $x_i = e^{y_i}$) and then logarithm the function. We start with the formal definition of a log-log transformation.

Definition 8.2.1 Let $f: \mathbb{R}^n_{>0} \to \mathbb{R}_{>0}$. A log-log transformation of the function f is the function $\tilde{f}: \mathbb{R}^n \to \mathbb{R}$ defined as

$$\tilde{f}(x_1,\ldots,x_n) = \log f\left(e^{x_1},\ldots,e^{x_n}\right).$$

The primary purpose of the log-log transformation is to convexify a function. In particular, we are interested in a family of functions that becomes convex after applying the log-log transformation.

Definition 8.2.2 Let $X \subseteq \mathbb{R}^n_{>0}$ be a convex set. A function $f: X \to \mathbb{R}$ is *log-log convex*, *log-log concave*, *log-log affine* if $f(x) > 0$ for all $x \in X$ and $\log f(e^{x_1},\ldots,e^{x_n})$ is convex, concave, affine respectively. Here the domain of the log-log transform is such that $(e^{x_1},\ldots,e^{x_n}) \in X$.

Similarly to log-convex functions it is possible to directly verify log-log convexity without explicitly performing the log-log transformation.

Theorem 8.2.3 A function $f: \mathbb{R}^n_{>0} \to \mathbb{R}_{>0}$ is log-log convex if and only if for all $x, y \in \mathbb{R}^n_{>0}$ and $\lambda \in [0,1]$ it holds that

$$f\left(x^\lambda \circ y^{1-\lambda}\right) \leq f(x)^\lambda \cdot f(y)^{1-\lambda}, \tag{8.3}$$

where $\circ$ is an element-wise product and powers are meant element-wise.

For log-log concave and log-log affine function the same statement holds with the '$\leq$' replaced by a '$\geq$' and '$=$' respectively.

Proof. We proceed similarly to the proof of Theorem 8.1.2. By definition f is log-log-convex if and only if $\log f(e^{x_1},\ldots,e^{x_n})$ is convex. Denote this function by g. By definition g is convex exactly if for all $a, b \in \mathbb{R}^n$ and $\lambda \in [0,1]$ we have

$$g(\lambda a + (1-\lambda)b) \leq \lambda g(a) + (1-\lambda) g(b). \tag{8.4}$$

Abbreviate $x_i = e^{a_i}, y_i = e^{b_i}$ and observe that

$$\begin{aligned} g(\lambda a + (1-\lambda)b) &= \log f(e^{\lambda a_1 + (1-\lambda)b_1}, \ldots, e^{\lambda a_n + (1-\lambda)b_n}) \\ &= \log f(x_1^\lambda y_1^{1-\lambda}, \ldots, x_n^\lambda y_n^{1-\lambda}) \\ &= \log f(x^\lambda \circ y^{1-\lambda}). \end{aligned}$$

Furthermore $g(a) = \log f(x)$. Plugging this into 8.4 gives us

$$\log f\left(x^\lambda \circ y^{1-\lambda}\right) \leq \log f(x)^\lambda + \log f(y)^{1-\lambda}.$$

After taking exponentials on both sides (which is an equivalent transformation as f is positive) we get the desired inequality. As a, b run over $\mathbb{R}^n$, x and y also run over all of $\mathbb{R}^n_{>0}$. So if (8.4) holds for all a, b then (8.3) holds for all x, y and vice versa, giving the desired equivalence.

For the cases of log-log concavity and log-log affinity the same prove holds after replacing (8.4) by the corresponding (in)equality for concave and affine functions.

■

■ **Example 8.2.4** Let $f:\ \mathbb{R}^n_{\geq 0} \to \mathbb{R}$ be a posynomial function, as defined in Definition 8.0.2, i.e., a function of the form

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{k1}} x_2^{a_{k2}} \cdots x_n^{a_{kn}}$$

where $c_k \in \mathbb{R}_{>0}$ and $a_k \in \mathbb{R}^n$ for all $k \in [K]$. The log-log transformation of f is the following

$$\log \sum_{k=1}^{K} c_k \exp\left(a_k^\top x\right),$$

that can be equivalently expressed in the following form

$$\log \sum_{k=1}^{K} \exp\left(a_k^\top x + b_k\right),$$

where $b_k = \log c_k$ for all $k \in [K]$. The log-log transformation of a posynomial function is a sum of exponentials of affine functions. ■

Many concepts from convex functions and convex sets can be directly translated into log-log convex setting. One of them is the epigraph of the function.

Definition 8.2.5 For the function $f:\ \mathbb{R}^n_{>0} \to \mathbb{R}_{>0}$ the set

$$\log \mathbf{epi} f := \{(\log x, \log t) \,|\, f(x) \leq t\}$$

is called the *log-log epigraph* of the function f. Here the $\log(x)$ is understood componentwise.

Similarly to an epigraph of a function f, there exists a strong connection between the log-log convexity of f and set convexity of $\log \mathbf{epi}$ of function f.

Theorem 8.2.6 A function $f:\ \mathbb{R}^n_{>0} \to \mathbb{R}_{>0}$ is log-log convex if and only if $\log \mathbf{epi} f$ is a convex set.

Proof. See Exercise 8.2. ■

We finalize this section with some examples of log-log convex and log-log concave functions.

■ **Example 8.2.7** Examples of log-log convex and log-log concave functions.

- *Sums.* $f:\mathbb{R}^n_{>0} \to \mathbb{R}_{>0}$ defined as $f(x) = \sum_{i=1}^n x_i$ is log-log convex.
- *Maximum.* $f:\mathbb{R}^n_{>0} \to \mathbb{R}_{>0}$ defined as $f(x) = \max\{x_i|\ i \in [n]\}$ is log-log convex.
- *Posynomials* are log-log convex.
- ℓ_p *norms* are log-log convex.
- $f: X \to \mathbb{R}_{>0}$ defined as $f(x) = x_1 - x_2$ is log-log concave for $X = \left\{ x \in \mathbb{R}^2_{>0} \mid x_1 > x_2 \right\}$.
- $f:(0,1) \to \mathbb{R}_{>0}$ defined as $f(x) = -x\log x$ is log-log concave.

■

8.3 Log-log convex program

Definition 8.3.1 Let f, g_i for all $i \in [m]$ be log-log convex functions and let h_j for all $j \in [\ell]$ be log-log affine functions. The *Log-Log Convex Program (LLCP)* in general form is the following

program:

$$\begin{aligned} &\min f(x) \\ &\quad g_i(x) \leq 1, \qquad \text{for } i \in [m], \\ &\quad h_j(x) = 1, \qquad \text{for } j \in [\ell], \\ &\quad x \in \mathbb{R}^n_{\geq 0}. \end{aligned} \tag{LLCP}$$

We are ready to prove the final ingredient to transform geometric programs into convex programs. We prove the following statement about the posynomial functions.

Proposition 8.3.2 Every posynomial function is a log-log convex function.

Proof. See Exercise 8.3 ■

With all the ingredients ready we present a geometric program in convex form.

Definition 8.3.3 Let

$$f = \sum_{k=1}^{K} c_k x_1^{a_{k1}} x_2^{a_{k2}} \cdots x_n^{a_{kn}}, \qquad g_i = \sum_{k=1}^{K_i} c_{ik} x_1^{a_{ik1}} x_2^{a_{ik2}} \cdots x_n^{a_{ikn}}$$

for all $i \in [m]$ be posynomial functions and let

$$h_j = \overline{c}_j x_1^{\overline{a}_{j1}} x_2^{\overline{a}_{j2}} \cdots x_n^{\overline{a}_{jn}}$$

for all $j \in [\ell]$ be monomial functions. Let $b_i = e^{c_i}$, $b_{i,k} = e^{c_{ik}}$ and $\overline{b}_j = e^{\overline{c}_j}$ The *Geometric Program (GP)* in convex form is the following program:

$$\begin{aligned} &\min \tilde{f}(x) = \log \sum_{k=1}^{K} \exp\left(a_k^\top x + b_k\right), \\ &\quad \tilde{g}_i(x) = \log \sum_{k=1}^{K_i} \exp\left(a_{ik}^\top x + b_{ik}\right) \leq 0, \qquad \text{for } i \in [m], \\ &\quad \tilde{h}_j(x) = \overline{a}_j^\top x + \overline{b}_j = 0, \qquad \text{for } j \in [\ell], \\ &\quad x \in \mathbb{R}^n_{\geq 0}. \end{aligned} \tag{GP$_{convex}$}$$

Here a_{ik} is the vector with entries $a_{ik1}, \ldots, a_{ikn}$.

We finalize the section with an important statement that a geometric program in convex form always gives the same optimal solution as the corresponding geometric program in general form.

Theorem 8.3.4 A geometric program in general form is feasible if and only if the corresponding geometric program in convex form is feasible. Moreover, for a feasible geometric program, an optimal solution for the geometric program in general form can be transformed to give an optimal solution for the geometric program in convex form, and vice versa.

Proof. We want to give a bijection between solutions of the geometric program

$$\begin{aligned}
&\min f(x)\\
&\quad g_i(x) \leq 1, \qquad \text{for } i \in [m],\\
&\quad h_j(x) = 1, \qquad \text{for } j \in [\ell],\\
&\quad x \in \mathbb{R}^n_{\geq 0}.
\end{aligned}$$

and (GP_{convex}) which respects the optimality value. For this let x be in $\mathbb{R}^n_{\geq 0}$ and set $y_i = \log x_i$ and $y = (y_1, \ldots, y_n)$. Then

$$e^{d^\top y + \alpha} = e^{d_1 y_1 + \cdots + d_n y_n + \alpha} = e^{d_1 \log x_1} \ldots e^{d_n \log x_n} e^{\alpha} = e^{\alpha} x_1^{d_1} x_2^{d_2} \ldots x_n^{d_n}.$$

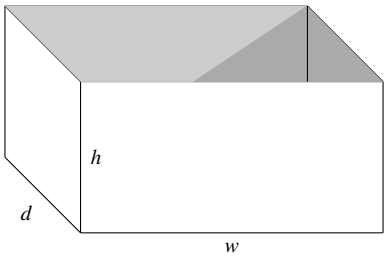
So plugging y into the constraints of (GP_{convex}) gives

$$\begin{aligned}
\tilde{f}(y) &= \log \sum_{k=1}^{K} \exp\left(a_k^\top y + b_k\right) = \log \sum_{k=1}^{K} e^{b_k} x_1^{a_{k1}} \ldots x_n^{a_{kn}} && = \log f(x),\\
\tilde{g}_i(y) &= \log \sum_{k=1}^{K_i} \exp\left(a_{ik}^\top y + b_{ik}\right) = \log \sum_{k=1}^{K_i} e^{b_{ik}} x_1^{a_{ik1}} \ldots x_n^{a_{ikn}} && = \log g_i(x), \qquad \text{for } i \in [m],\\
\tilde{h}_j(y) &= \overline{a}_j^\top y + \overline{b}_j = \log\left(e^{\overline{b}_j} x_1^{\overline{a}_{j1}} \ldots x_n^{\overline{a}_{jn}}\right) && = \log h(x), \qquad \text{for } j \in [\ell].
\end{aligned}$$

So x satisfies the constraints of the geometric program if and only if y satisfies the constraints of (GP_{convex}). Furthermore the objective value of y is the logarithm of the objective value of x, giving the correspondence of optimal solutions. The other direction works the same as given y we can construct x as $x_i = e^{y_i}$. So the solutions of both programs are in an objective conserving bijection, proving the claim. ■

8.4 Applications

■ **Example 8.4.1** Consider the task of designing a box. The walls are of a different material than the bottom and on both usages there is a constraint how much material is available. There is no top part. At the same time for stability reasons you might want the box to be no higher than either side of the base. You want to satisfy these constraints with a box with maximal volume.



This can be modeled as a geometric program. For this let h, d, w denote the height, depth and width of the box. The material constraints give upper bounds on the sizes of $2(hd + hw) \leq M_{side}$ and $wd \leq M_{base}$ where M_{side} and M_{base} denote the amount of the two materials available. The stability constraint can be modeled as $h \leq w$ and $h \leq d$. Finally we want to maximize the volume which is hdw so we can write the problem as:

$$
\begin{aligned}
\max\ & hdw \\
& 2(hd+hw) \le M_{side} \\
& wd \le M_{base} \\
& h \le w \\
& h \le d \\
& h,d,w \ge 0.
\end{aligned}
$$

This almost is a geometric program, the constraints already have the right form but a GP is a minimization problem. This can easily be achieved by noticing that hdw is maximal exactly when $h^{-1}d^{-1}w^{-1}$ is minimal, so the final geometric program would be:

$$
\begin{aligned}
\min\ & h^{-1}d^{-1}w^{-1} \\
& 2(hd+hw) \le M_{side} \\
& wd \le M_{base} \\
& h \le w \\
& h \le d \\
& h,d,w \ge 0.
\end{aligned}
$$

■

8.5 Exercises

Exercise 8.1 Prove that:

1. the set of monomial functions is closed under multiplcation and division,
2. the set of posynomial functions is closed under addition and multiplication.

Exercise 8.2 Prove Theorem 8.2.6 that f is a log-log convex function if and only if $\log \mathbf{epi} f$ is a convex set.

Exercise 8.3 Prove Proposition 8.3.2 that every posynomial function is a log-log convex function.

Exercise 8.4 In this exercise we prove a composition rule for log-log convex/concave functions. A similar statement for convex/concave functions was proved in Exercise 1.3.

Let f be log-log convex function. Prove that $f(g(x))$ is log-log convex in the following cases:

1. f is nondecreasing and g is log-log convex,
2. f is nonincreasing and g is log-log concave,
3. g is log-log affine.

Exercise 8.5 Let $X \subseteq \mathbb{R}^n_{>0}$ be a convex set, $f_1,\ f_2 \colon X \to \mathbb{R}_{>0}$ and $g_1,\ g_2 \colon X \to \mathbb{R}_{>0}$ be log-log convex and log-log concave functions, respectively. Let f and $g \colon X \to \mathbb{R}_{>0}$ be a convex and concave function, respectively. For the following statements decide whether they are true and give a short proof or counterexample respectively.

i) $f_1 + f_2$ is log-log convex.
ii) $g_1 + g_2$ is log-log concave.
iii) f is log-log convex.
iv) g is log-log concave.

Exercise 8.6 Decide which of the following functions $f_i \colon \mathbb{R} \to \mathbb{R}_{>0}$ are log-convex

i) $f_1(x) = \log(p) + |x|^p$ for $p \in [2, \infty)$
ii) $f_2(x) = \exp(|x|^p)$ for $p \in [1, \infty]$
iii) $f_3(x) = \frac{1}{|x|^p}$ for $x \in (0, \infty)$ and $p > 0$

Decide which of the following functions $f_i \colon \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ are log-log convex

i) $f_1(x) = \max\left\{x, \frac{1}{x}\right\}$
ii) $f_2(x) = \exp\left(\frac{(x-\mu)^2}{\sigma^2}\right)$ for $\mu \in \mathbb{R}$ and $\sigma > 0$.
iii) $f_3(x) = \tan^{-1}(x)$

Exercise 8.7 Prove that the gamma function $\Gamma \colon \mathbb{R}_{>0} \to \mathbb{R}$,

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$$

is log convex.

Hint: You may use that the euler form of the Gamma function

$$\Gamma(z) = \lim_{m \to \infty} \frac{m^z m!}{z(z+1)\cdot\cdots\cdot(z+m)}, \qquad z > 0$$

is analytical.

Exercise 8.8 — Programming Exercise. In this exercise we want to see how to use CVXPY to solve a GP. It works more or less the same way. First specify the program by defining the objective

and constraint functions. Then call the `solve` function, but now with the `problem.solve(gp=True)` option.

Consider the following problem: We are given a matrix $A \in \mathbb{R}^{n\times m}$ and want to approximate it as closely as possible by a rank one matrix $x^\top \cdot y$. Here we use the following metric: For each entry we calculate the relative derivation given by $R(a,b) = \max\{a/b, b/a\} - 1$. In the end we want to minimize the sum of all relative derivations.

Formulate this as a convex optimization problem and implement it in python.

8.6 Solutions

Solution of the Exercise 8.1

1. Consider two monomials $f(x) = cx_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}$ and $g(x) = dx_1^{b_1}x_2^{b_2}\cdots x_n^{b_n}$. Their product and quotient can be written as monomials as follows:

$$(fg)(x) = (cd)x_1^{a_1+b_1}x_2^{a_2+b_2}\cdots x_n^{a_n+b_n}$$

and

$$\frac{f}{g}(x) = \frac{c}{d}x_1^{a_1-b_1}x_2^{a_2-b_2}\cdots x_n^{a_n-b_n}.$$

2. Consider two posynomials $f(x) = \sum_{k=1}^{K} c_k x_1^{a_{k1}}x_2^{a_{k2}}\cdots x_n^{a_{kn}}$ and $g(x) = \sum_{k=1}^{K'} d_k x_1^{a'_{k1}}x_2^{a'_{k2}}\cdots x_n^{a'_{kn}}$. Their sum can be written as a posynomial

$$(f+g)(x) = \sum_{k=1}^{L} d_k x_1^{b_{k1}}x_2^{b_{k2}}\cdots x_n^{b_{kn}}$$

where $L = K + K'$ and $b_{ki} = \begin{cases} a_{ki} & \text{if } k \le K, \\ a'_{ki} & \text{if } k > K \end{cases}$ and analogously $d_k = \begin{cases} c_k & \text{if } k \le K, \\ c'_k & \text{if } k > K. \end{cases}$ Their product can be written as the posynomial

$$\sum_{k=1}^{K}\sum_{l=1}^{K'}(c_k d_l)x_1^{a_{k1}+a'_{l1}}x_2^{a_{k2}+a'_{l2}}\cdots x_n^{a_{kn}+a'_{ln}}.$$

In other words we could reason that the product of two posynomials is a sum of products of monomials which by the first part is a sum of monomials, hence a posynomial.

Solution of the Exercise 8.2

We will show that $\log \mathbf{epi} f$ is the epigraph of the log-log transformation of f. Then we are done by Proposition 1.1.10 which says that a function is convex if and only if its epigraph is convex.

Let $\tilde{f}$ be the log-log transformation of f. We have to show that a point (y,s) lies in $\log \mathbf{epi} f$ if and only if it lies in $\mathbf{epi}\tilde{f}$. We have by definition $(y,s) \in \log \mathbf{epi} f$ if there are x and t with $f(x) \le t$ and $(y,s) = (\log x, \log t)$. By substituting $x_i = e^{y_i}$ and $t = e^s$ we have $(y,s) \in \log \mathbf{epi} f$ if and only if $f(e^{y_1},\ldots,e^{y_n}) \le e^s$. This is equivalent to $\log f(e^{y_1},\ldots,e^{y_n}) \le s$. Note that the left hand side is exactly the log-log transformation $\tilde{f}$ of f, so the inequality classifies exactly the points in the epigraph of $\tilde{f}$, finishing the proof.

Solution of the Exercise 8.3

Let $f = \sum_{k=1}^{K} c_k x_1^{a_{1k}}x_2^{a_{2k}}\cdots x_n^{a_{nk}}$ denote a posynomial. We will use the classification of Theorem 8.2.3. So let $x, y \in \mathbb{R}^n$ be given as well as $\lambda \in [0,1]$. Then we have:

$$\begin{aligned} f(x^\lambda \circ y^{1-\lambda}) &= \sum_{k=1}^{K} c_k (x_1^\lambda y_1^{1-\lambda})^{a_{1k}} (x_2^\lambda y_2^{1-\lambda})^{a_{2k}} \cdots (x_n^\lambda y_n^{1-\lambda})^{a_{nk}} \\ &= \sum_{k=1}^{K} (c_k x_1^{a_{1k}}x_2^{a_{2k}}\cdots x_n^{a_{nk}})^\lambda (c_k y_1^{a_{1k}}y_2^{a_{2k}}\cdots y_n^{a_{nk}})^{1-\lambda}. \end{aligned}$$

Now we can apply Hölder's inequality. It states that for two sequences of real numbers $v_1,\ldots,v_m$ and $w_1,\ldots,w_m$ as well as a $\lambda \in [0,1]$ we have $\sum_{k=1}^{m} v_k^\lambda w_k^{1-\lambda} \le (\sum_{k=1}^{m} v_k)^\lambda (\sum_{k=1}^{m} w_k)^{1-\lambda}$. Applying it above bounds the final expression from above by

$$\le \left(\sum c_k x_1^{a_{1k}}x_2^{a_{2k}}\cdots x_n^{a_{nk}}\right)^\lambda \left(\sum c_k y_1^{a_{1k}}y_2^{a_{2k}}\cdots y_n^{a_{nk}}\right)^{1-\lambda} = f(x)^\lambda f(y)^{1-\lambda}.$$

Altogether it shows the desired inequality, hence f is log-log convex.

Solution of the Exercise 8.4

Consider the log-log transformation of $f \circ g$. It can be written as

$$\log f(g(e^x)) = \log f(e^{\log g(e^x)}),$$

which is the composition of the log-log transformations of f and g. Now we can use the composition rules for usual convex functions, Proposition 1.1.11. In all three cases given we can conclude from convexity of the log-log transformation of f together with the condition on the log-log transformation of g that the composition of these two transformations is convex, hence $f \circ g$ is log-log convex.

Alternative solution:

We want to use the equivalent characterization of log-log convexity given in Theorem 8.2.3. So we know that for all $x, y \in X$ and $\lambda \in [0,1]$ it holds that

$$f\left(x^\lambda \circ y^{1-\lambda}\right) \leq f(x)^\lambda \cdot f(y)^{1-\lambda}.$$

Further we have that one of the following holds:

1. f is nondecreasing and $g\left(x^\lambda \circ y^{1-\lambda}\right) \leq g(x)^\lambda \cdot g(y)^{1-\lambda}$,
2. f is nonincreasing and $g\left(x^\lambda \circ y^{1-\lambda}\right) \geq g(x)^\lambda \cdot g(y)^{1-\lambda}$,
3. $g\left(x^\lambda \circ y^{1-\lambda}\right) = g(x)^\lambda \cdot g(y)^{1-\lambda}$.

In the first two cases we can conclude:

$$f\left(g\left(x^\lambda \circ y^{1-\lambda}\right)\right) \leq f\left(g(x)^\lambda \cdot g(y)^{1-\lambda}\right) \leq f\left(g(x)^\lambda\right) \cdot f\left(g(y)^{1-\lambda}\right).$$

In the third case we have

$$f\left(g\left(x^\lambda \circ y^{1-\lambda}\right)\right) = f\left(g(x)^\lambda \cdot g(y)^{1-\lambda}\right) \leq f\left(g(x)^\lambda\right) \cdot f\left(g(y)^{1-\lambda}\right).$$

So in all three cases we have shown that $f \circ g$ is log-log convex, as desired.

Solution of the Exercise 8.5

i) True: This follows from Theorem 8.2.3 with the same arguments as in the proof of Proposition 8.1.3 5.

ii) False: Note that if $f\colon \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ is log convex / concave, then $f \circ \log\colon \mathbb{R}_{>1} \to \mathbb{R}_{>0}$ is log-log convex / concave. Therefore the counterexample from Proposition 8.1.3 5 can be used again.

iii) False: Consider $f = e^{-x}\colon \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ which is convex. Then $\log(f(e^x)) = -e^x$ which is concave.

iv) TODO

Solution of the Exercise 8.6

Starting with the log-convex functions.

i) False: Already for $x > 0$ we get that

$$\log(f_1(x)) = \log(C + x^p)$$

where C is independent of x and hence a nonconvex function.

ii) True: Immediately follows from the convexity of $|x|^p$ for $p \geq 1$.

iii) True: We get

$$\log(f_3(x)) = -p\log(x)$$

which is convex for $x > 0$.

Now for the log-log convexity.

i) True: $\log(f_1(e^x)) = f_1(x)$ due to monotonicity of the logarithm and hence convex.

ii) False: Calculating yields

$$\log(f_2(e^x)) = \frac{(e^x-\mu)^2}{\sigma^2}$$

with second derivative

$$(\log(f_2(e^x)))'' = \frac{4e^{2x}-2e^x\mu}{\sigma^2} = \frac{2e^x}{\sigma^2}(2e^x-\mu).$$

This is convex on $[\log\left(\frac{\mu}{2}\right),\infty)$ hence convex on its domain only if $\mu \geq 2$.

iii) False: This can either be seen by plugging in well known values of tan, e.g.

$$x \in \left\{\log\left(\sqrt{3}/3\right), \log(1), \log\left(\sqrt{3}\right)\right\},$$

or differentiating. Here the (more complicated) differentiation way:

$$\begin{aligned}
\log(f_3(x)) &= \log\left(\tan^{-1}(e^x)\right) \\
\frac{d}{dx}\log\circ\tan^{-1}\circ\exp(x) &= \frac{e^x}{(e^{2x}+1)\tan^{-1}(e^x)} \\
\frac{d^2}{dx^2}\log\circ\tan^{-1}\circ\exp(0) &= -\frac{1}{4\tan^{-1}(1)^2} \\
&= -\frac{4}{\pi^2} < 0.
\end{aligned}$$

Solution of the Exercise 8.7

As stated in the hint we will use the euler form

$$\Gamma(z) = \lim_{m\to\infty} \frac{m^z m!}{z(z+1)\cdot\dots\cdot(z+m)}$$

for $z > 0$. Now we will show that the second derivative is strictly positive, implying convexity. We use the hint that Γ is analytical in order to justify the following limit exchange

$$\begin{aligned}
\frac{d^2\log\circ\Gamma}{dz^2}(z) &= \frac{d^2}{dz^2}\log\left(\lim_{m\to\infty}\frac{m^z m!}{z(z+1)\cdot\dots\cdot(z+m)}\right) \\
&= \lim_{m\to\infty}\frac{d^2}{dz^2}\log\left(\frac{m^z m!}{z(z+1)\cdot\dots\cdot(z+m)}\right).
\end{aligned}$$

Now we calculate for a fixed $m \in \mathbb{N}$

$$\begin{aligned}
\frac{d^2}{dz^2}\log\left(\frac{m^z m!}{z(z+1)\cdot\dots\cdot(z+m)}\right) &= \frac{d^2}{dz^2}\left(z\log(m)+\log(m!)-\sum_{n=0}^{m}\log(z+n)\right) \\
&= \sum_{n=0}^{m}\frac{1}{(z+n)^2}.
\end{aligned}$$

This gives

$$\begin{aligned}\frac{d^2 \log \circ \Gamma}{dz^2}(z) &= \lim_{m\to\infty} \sum_{n=0}^{m} \frac{1}{(z+n)^2} \\ &= \sum_{n=0}^{\infty} \frac{1}{(z+n)^2} > 0,\end{aligned}$$

finishing the proof.

IV Appendix

9. Mathematical background

This book assumes some level of mathematical maturity of the reader. In the appendix we provide basic mathematical facts that we assume reader should now before studying this book. For readers that are not familiar with mathematical background used in the main body of the book we recommend studying the Appendix first.

9.1 Linear Algebra

We start with some basic properties of matrices. Therefore we denote the set of all real $m \times n$ matrices as $\mathbb{R}^{m\times n}$ for $m,n \in \mathbb{N}_{\geq 1}$,.

We start with one of the most important definitions for matrices.

Definition 9.1.1 — Eigenvalue. Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix and $\lambda \in \mathbb{R}$. Then λ is called an *eigenvalue* of A, if there exists a $v \in \mathbb{R}^n$ such that

$$Av = \lambda v.$$

In this case v is called an *eigenvector* for the eigenvalue λ.

It's of special interest to look at matrices that have nonnegative eigenvalues. Therefore the following definition will be used often throughout this book.

nonnegative eigenvalues

Definition 9.1.2 — Positive (semi-)definite. Let $A \in \mathbb{R}^{n\times n}$ be a quadratic matrix. Then A is *positive semidefinite* (PSD), if

are important for the problems in this book

$$\forall x \in \mathbb{R}^n \setminus \{0\} : x^\top Ax \geq 0.$$

We say A is *positive definite* if the above inequality holds strictly, i.e.

$$\forall x \in \mathbb{R}^n \setminus \{0\} : x^\top Ax > 0.$$

To bring this property in connection with eigenvalues we first introduce some notation.

Definition 9.1.3 Let $n \in \mathbb{N}_{\geq 1}$. Then we denote the set of all $n \times n$ symmetric matrices as $\mathcal{S}^n$, i.e.

$$\mathcal{S}^n = \left\{ A \in \mathbb{R}^{n \times n} \mid A^\top = A \right\}.$$

Furthermore we denote the *positive semidefinite cone* as

$$\mathcal{S}^n_+ = \{ A \in \mathcal{S}^n \mid A \succeq 0 \}$$

where $A \succeq 0$ iff A is symmetric and positive semidefinite. For symmetric positive definite matrices we use the notations $A \succ 0$ and $\mathcal{S}^n_{++}$.

Now we finally get the following connection.

Proposition 9.1.4 For any $A \in \mathcal{S}^n$, the following statements are equivalent:

1. The matrix A is positive semidefinite, i.e., $A \in \mathcal{S}^n_+$.
2. For all $x \in \mathbb{R}^n$ we have $x^\top A x \geq 0$.
3. All eigenvalues of A are nonnegative.
4. All principal minors of A are nonnegative.
5. There exists a factorization $A = B^\top B$ called *Cholesky decomposition*, where B is a triangular matrix.

Proof. Exercise 7.1. ■

While the Cholesky Decomposition only exists for PSD matrices, general symmetric matrices can also be decomposed in a useful manner.

Theorem 9.1.5 — Spectal Theorem for symmetric Matrices. Let $A \in \mathcal{S}^n$ be a symmetric matrix. Then there exist eigenvectors

$$v_1, \ldots, v_n \in \mathbb{R}^n$$

such that A can be written as

$$A = VDV^\top,$$

where

$$V = \begin{pmatrix} | & & | \\ v_1 & \ldots & v_2 \\ | & & | \end{pmatrix}$$

is the matrix with columns v_i and $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ is the diagonal-matrix with A's eigenvalues on its diagonal. In particular A has all n eigenvalues.

Since general (nonsymmetric) real matrices do not necessarily have n eigenvalues, the following definition is useful in many cases. Furthermore it is also applicable to nonquadratic matrices.

Definition 9.1.6 — Orthogonal Matrices. Let $n \in \mathbb{N}_{\geq 1}$ and $V \in \mathbb{R}^{n \times n}$. Then V is called *orthogonal* if $V^\top V = I_n$. We use the notation $O(n)$ to denote the set of all orthogonal $n \times n$ matrices.

Theorem 9.1.7 — Singular Value Decomposition. Let $A \in \mathbb{R}^{m \times n}$ be a (not necessarily quadratic) matrix. Then there exist orthogonal matrices $U \in O(m), V \in O(n)$ and a diagonal matrix Σ such that

$$A = U \Sigma V^\top.$$

Here Σ has the form

$$\Sigma = \left(\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_r & \\ \hline & 0 & & 0 \end{array}\right) \in \mathbb{R}^{m\times n}$$

and $\sigma_1, \dots, \sigma_r$ are called the singular values of A.

By the proof of Theorem 9.1.7 (which we omit) it follows that $\sigma_i(A) = \sqrt{\lambda_i(A^\top A)}$ where we often sort $\sigma_1 \geq \dots \sigma_r$.

Corollary 9.1.8 Let $A \in \mathscr{S}^n_+$. Then we have for $i \in [n]$

$$\sigma_i(A)^2 = \lambda_i(A)$$

after ordering $\lambda_1 \geq \dots \geq \lambda_n$.

Proof. Since A is PSD we can write $A = VDV^\top$ with $D = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$. Then we get

$$\begin{aligned} A^\top A &= \left(VDV^\top\right)^\top VDV^\top \\ &= VD^\top V^\top VDV^\top \\ &= VD^\top DV^\top \\ &= VDDV^\top. \end{aligned}$$

Since $DD = \operatorname{diag}\left(\lambda_1^2, \dots, \lambda_n^2\right)$ we see that $A^\top A$ has the eigenvalues $\lambda_1^2, \dots, \lambda_n^2$ which implies the statement. ■

Finally we derive a way to decide whether certain types of matrices are positive (semi-)definite via the so called *Schur complement*. Therefore we will partition matrices $X \in \mathscr{S}^n$ in the form

$$X = \left(\begin{array}{c|c} A & B \\ \hline B^\top & C \end{array}\right).$$

Definition 9.1.9 — Schur complement. Let $X \in \mathscr{S}^n$ and suppose the A in the partition above is invertible. Then

$$S_A := C - B^\top A^{-1} B$$

is called *Schur complement* of A in X. Analogously we define the *Schur complement* of C in X as

$$S_C := A - BC^{-1}B^\top.$$

Now we can characterize when X is positive (semi-)definite.

Proposition 9.1.10 Let $X \in \mathscr{S}^n$ and suppose A is invertible. Then

i) $X \succ 0 \Leftrightarrow A \succ 0$ and $S_A \succ 0$,

ii) If $A \succ 0$, then $X \succeq 0 \Leftrightarrow S_A \succeq 0$.

On the other hand, if C is invertible we get

iii) $X \succ 0 \Leftrightarrow C \succ 0$ and $S_C \succ 0$,

iv) If $C \succ 0$, then $X \succeq 0 \Leftrightarrow S_C \succeq 0$.

In order to also deal with noninvertible diagonal matrices we need to introduce further notation.

Definition 9.1.11 — Pseudo-Inverse. Let $A \in \mathbb{R}^{m\times n}$ with $r := \operatorname{rank}(A)$. Further let

$$A = U\Sigma V^\top$$

be a SVD of A. Then the *pseudo-invers* of A is defined as

$$A\dagger := U\Sigma^\dagger V^\top$$

where

$$\Sigma^\dagger := \operatorname{diag}\sigma_1^{-1},\ldots,\sigma_r^{-1},0,\ldots,0.$$

It can be proven that *the* pseudo-invers does not depend on the choice of U,V, justifying the use of the article *the*.

Proposition 9.1.12 Let $X \in \mathscr{S}^n$ and define the partition

$$X =: \left(\begin{array}{c|c} A & B \\ \hline B^\top & C \end{array}\right).$$

Then the following statements are equivalent.

i) $X \succeq 0$,
ii) $A \succeq 0$ and $\left(I - AA^\dagger\right)B = 0$ and $S_A^\dagger := C - B^\top A^\dagger B \succeq 0$,
ii) $C \succeq 0$ and $\left(I - CC^\dagger\right)B = 0$ and $S_C^\dagger := A - B^\top C^\dagger B \succeq 0$.

9.2 Differentiability

We start with the definition of a differentiable function in one variable.

Definition 9.2.1 — univariate differentiable function. A function $f : U \to \mathbb{R}$, for $U \subseteq \mathbb{R}$ being an open set, is *differentiable* at a point $\bar{x} \in U$ if the derivative

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

exists. If the function is differentiable at every point $\bar{x} \in U$, the function is *differentiable* on U.

We can extend the definition of differentiability for function of several variables.

Definition 9.2.2 — multivariate differentiable function. A function $f : U \to \mathbb{R}$, for $U \subseteq \mathbb{R}^n$ being an open set, is *differentiable* at a point $\bar{x} \in U$ if there exists a linear function $\mathbb{J} \in \mathbb{R}^{1\times n}$ such that

$$\lim_{h\to 0} \frac{\|f(\bar{x}+h) - f(\bar{x}) - \mathbb{J}h\|_{\mathbb{R}^n}}{\|h\|_{\mathbb{R}^n}} = 0,$$

where $\|\cdot\|_{\mathbb{R}^n}$ is a norm in $\mathbb{R}^n$. In this case we call $\mathbb{J}(\bar{x})$ the Jacobian of f at $\bar{x}$. If the function is differentiable at every point $\bar{x} \in U$ the function is *differentiable* on U.

Definition 9.2.3 — gradient. The *gradient* of a differentiable function $f : U \subseteq \mathbb{R}^n \to \mathbb{R}$ at a point $\bar{x} = (\bar{x}_1,\ldots,\bar{x}_n) \in S$ is defined as

$$\nabla f(\bar{x}) = \mathbb{J}(\bar{x})^\top \in \mathbb{R}^n.$$

A function is called *continuously differentiable* on the set U if it is differentiable and its gradient is continuous on the set U. For convex functions the properties of being differentiable and continuously differentiable conveniently coincide. The proof does however need further properties and hence is omitted.

Proposition 9.2.4 A differentiable convex function $f: U \subseteq \mathbb{R}^n \to \mathbb{R}$, for U being an open, convex set, is continuously differentiable.

An intuitive explanation of a differentiable function is that it can be locally approximated by linear functions. More precisely, a function $f: U \to \mathbb{R}$ is differentiable at a point $\bar{x} \in U$ if there exist

- a gradient vector $\nabla f(\bar{x}) \in \mathbb{R}^n$,
- and a function $\alpha_{\bar{x}}: \mathbb{R}^n \to \mathbb{R}$ with $\lim_{v \to 0} \alpha_{\bar{x}}(v) = 0$,

such that

$$f(\bar{x}+\delta) = f(\bar{x}) + \nabla f(\bar{x})^\top \delta + \|\delta\|_2 \, \alpha_{\bar{x}}(\delta),$$

for all $\delta \in \mathbb{R}^n$ with $\bar{x}+\delta \in U$.

The intuition can be extended to twice differentiable functions, for which local approximation can be done with quadratic functions. Namely, a function $f: U \to \mathbb{R}$ is twice differentiable at a point $\bar{x} \in U$ if there exist

- a gradient vector $\nabla f(\bar{x}) \in \mathbb{R}^n$,
- a function $\alpha_{\bar{x}}: \mathbb{R}^n \to \mathbb{R}$ with $\lim_{v \to 0} \alpha_{\bar{x}}(v) = 0$,
- and a Hessian matrix $H(\bar{x}) = \nabla^2 f(\bar{x}) \in \mathbb{R}^{n \times n}$,

such that

$$f(\bar{x}+\delta) = f(\bar{x}) + \nabla f(\bar{x})^\top \delta + \frac{1}{2}\delta^\top H(\bar{x})\delta + \|\delta\|_2^2 \, \alpha_{\bar{x}}(\delta),$$

for every $\delta \in \mathbb{R}^n$ with $\bar{x}+\delta \in U$.

In fact the above intuitive explanation leads to the following statements about approximation of convex function on a convex open set.

Theorem 9.2.5 Let $f: U \to \mathbb{R}$, for U being a convex open set, be a convex differentiable function. Then for any $x, y \in U$ we have

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

Proof. Since f is convex we get for all $x, y \in U$ and $t \in (0,1)$

$$\begin{aligned} f(x+t(y-x)) &= f(ty+(1-t)x) \\ &\leq t f(y) + (1-t) f(x) \\ &= f(x) + t\,(f(y) - f(x)). \end{aligned}$$

dividing by t and reordering yields

$$f(y) \geq f(x) + \frac{f(x+t(y-x)) - f(x)}{t}.$$

If we now use the characterisation of differentiable functions for $\bar{x} = x, \delta = t(y-x)$ we get

$$\begin{aligned} f(y) &\geq f(x) + \frac{\nabla f(x)^\top t(y-x) + \|t(y-x)\|_2 \, \alpha_x(t(y-x))}{t} \\ &= f(x) + \nabla f(x)^\top (y-x) + \|(y-x)\|_2 \, \alpha_x(t(y-x)). \end{aligned}$$

Since $\|y-x\|_2 \, \alpha_x(t(y-x)) \xrightarrow{t \to 0} 0$ by assumption, we get the claim. ∎

The function $\alpha_{\bar{x}}$ from before can be more precisely characterized. This is done using the important taylor theorem. We will only state the general theorem for one dimension and the first and second order in multiple dimensions since more general forms require a lot of notation.

Theorem 9.2.6 — Univariate Taylor's Theorem. Let $I \subseteq \mathbb{R}$ be an open interval and let $f\colon I \to \mathbb{R}$ be $(n+1)$-times continuously differentiable function. Then for every $x \in I$ and $\delta \in \mathbb{R}$ with $(x+\delta) \in I$ there exists $\xi_\delta \in \mathbb{R}$ with $|x-\xi_\delta| \leq \delta$ such that

$$f(x+\delta) = \sum_{k=0}^{n} \frac{f^{(k)}(x)}{k!}\delta^k + R_{n+1}(\delta),$$

where

$$R_{n+1}(\delta) = \frac{f^{(n+1)}(\xi_\delta)}{(n+1)!}.$$

R The function $\alpha_{\bar{x}}$ is then given by

$$\alpha_{\bar{x}}(\delta) = \frac{f^{(n+1)}(\xi_\delta) - f^{(n+1)}(\delta)}{(n+1)!}.$$

The proof is a basic induction but will be omitted since it requires some results which would take too long to derive in a self contained manner.

For the multidimensional case we restrict ourself to the first and second order statement as mentioned before. Those statements suffice to prove many results in optimization.

Theorem 9.2.7 — First order multidimensional Taylor. Let $U \subseteq \mathbb{R}^n$ be an open set and let $f\colon U \to \mathbb{R}$ be a continuously differentiable function. Then for any $x, y \in U$ there exists $z \in [x,y]$ such that

$$f(y) = f(x) + \nabla f(z)^\top (y-x).$$

Here $[x,y]$ denotes the line segment defined in Definition 1.1.1.

Theorem 9.2.8 — Second order multidimensional Taylor. Let $U \subseteq \mathbb{R}^n$ be an open set and let $f\colon U \to \mathbb{R}$ be a twice continuously differentiable function. Then for any $x, y \in U$ there exists $z \in [x,y]$ such that

$$f(y) = f(x) + \nabla f(x)^\top (y-x) + \frac{1}{2}(y-x)^\top H(z)(y-x).$$

Proof. Both statements follow by looking at the function

$$\Phi(t) = f(x + t(y-x))$$

and applying the univariate Taylor Theorem on g around 0. Therefore we realize that $f(x) = \Phi(0)$ and $f(y) = \Phi(1)$. Calculating

$$\Phi'(t) = \nabla f(x + t(y-x))^\top (y-x)$$

and

$$\Phi''(t) = (y-x)^\top H(x + t(y-x))(y-x).$$

then gives the statements in the form (here for the second order case)

$$\Phi(1) = \Phi(0) + \Phi'(0) + \frac{1}{2}\Phi''(z)$$

where $z = x + \tau(y-x)$ and $\tau \in [0,1]$. ∎

Finally, we can present necessary and sufficient conditions for optimality of differentiable functions.

Proposition 9.2.9 — First order Necessary Optimality Condition. Let $f\colon U \to \mathbb{R}$, for U being an open set, be a differentiable function and let $\bar{x}$ be a local extrema of f (i.e. a local minimum or local maximum), then $\nabla f(\bar{x}) = 0$.

Proof. This statement again follows from the 1-dimensional equivalent which is proven in high school. Therefore let $i \in [n]$ and $\Phi(t) = f(\bar{x} + te_i)$ for small enough t. Then Φ has a global extrema at 0 as well and we get

$$\nabla f(\bar{x})_i = \Phi'(0) = 0$$

proving the statement. ■

In high school the second order sufficient condition is taught for the univariate case. It informally states that if $f'(x) = 0$ and $f''(x) > 0$ then x is a local minima. To transport this statement to higher dimensions we use the notion of matrices being positive definite as defined in Definition 9.1.2.

Proposition 9.2.10 — Second order Sufficient Optimality Condition. Let $f\colon U \to \mathbb{R}$, for U being an open set, be a twice differentiable function and let $\bar{x}$ be a point at which $\nabla f(\bar{x}) = 0$. If $H(\bar{x})$ is positive definite then $\bar{x}$ is a local minimum.

Proof. This time we use our characterization

$$f(\bar{x}+\delta) = f(\bar{x}) + \nabla f(\bar{x})^\top \delta + \frac{1}{2}\delta^\top H(\bar{x})\delta + \|\delta\|_2^2 \alpha_{\bar{x}}(\delta),$$

for every $\delta \in \mathbb{R}^n$ with $\bar{x} + \delta \in U$. Let

$$0 < \alpha \coloneqq \min_{\|h\|_2 = 1} h^\top H(\bar{x}) h$$

then we get

$$\begin{aligned} f(\bar{x}-\delta) - f(\bar{x}) &= \frac{\|\delta\|^2}{2}\left(\left(\frac{\delta}{\|\delta\|}\right)^\top H(\bar{x})\left(\frac{\delta}{\|\delta\|}\right) + 2\alpha_{\bar{x}}(\delta)\right) \\ &\geq \frac{\|\delta\|^2}{2}\left(\alpha + 2\alpha_{\bar{x}}(\delta)\right). \end{aligned}$$

Since $\alpha_{\bar{x}}(\delta) \xrightarrow{\delta \to 0}$ we can find $d > 0$ such that $\alpha > 2|\alpha_{\bar{x}}(\delta)|$ for all $\|\delta\| < d$. This gives us that $\bar{x}$ is a local minimum in the ball with radius d centred at $\bar{x}$ intersected with U. ■

Bibliography

[1] Egon Balas. “Disjunctive Programming: Cutting Planes From Logical Conditions”. In: *Nonlinear Programming 2*. Edited by O.L. Mangasarian, R.R. Meyer, and S.M. Robinson. Academic Press, 1975, pages 279–312 (cited on page 32).

[2] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004 (cited on pages 12, 29, 36).

[3] Siu On Chan et al. “Approximate Constraint Satisfaction Requires Large LP Relaxations”. In: *J. ACM* 63.4 (2016), 34:1–34:22 (cited on page 170).

[4] Moses Charikar, Konstantin Makarychev, and Yury Makarychev. “Integrality gaps for Sherali-Adams relaxations”. In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC*. 2009, pages 283–292 (cited on page 170).

[5] George Bernard Dantzig. *Linear Programming and Extensions*. Santa Monica, CA: RAND Corporation, 1963. DOI: 10.7249/R366 (cited on page 18).

[6] Charles Delorme and Svatopluk Poljak. “Laplacian eigenvalues and the maximum cut problem”. In: *Math. Program.* 62 (1993), pages 557–574 (cited on page 170).

[7] M.R. Garey and D.S. Johnson. ““Strong” NP-completeness results: motivation, examples, and implications”. In: *Journal of the ACM* 25 (1978), pages 499–508 (cited on page 168).

[8] Michel X. Goemans and David P. Williamson. “Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming”. In: *J. ACM* 42.6 (1995), pages 1115–1145 (cited on pages 168, 170).

[9] Peter Gritzmann. *Lecture Notes on Linear and Convex Optimization, Technical University of Munich*. Nov. 2020 (cited on page 47).

[10] Marshall Hall. *Combinatorial Theory (2nd Ed.)* USA: John Wiley & Sons, Inc., 1998. ISBN: 0471315184 (cited on page 180).

[11] Johan Håstad. “Clique is Hard to Approximate Within n$^{1\text{-epsilon}}$”. In: *37th Annual Symposium on Foundations of Computer Science, FOCS ’96, Burlington, Vermont, USA, 14-16 October, 1996*. 1996, pages 627–636 (cited on page 58).

[12] J. William Helton and Jiawang Nie. “Sufficient and Necessary Conditions for Semidefinite Representability of Convex Hulls and Sets”. In: *SIAM J. Optim.* 20.2 (2009), pages 759–791 (cited on page 178).

[13] Leonid Genrikhovich Khachiyan. “A polynomial algorithm in linear programming”. In: *Doklady Akademii Nauk*. Volume 244. 5. Russian Academy of Sciences. 1979, pages 1093–1096 (cited on page 19).

[14] V. Klee and G. Minty. “How good is the simplex algorithm?” In: 1970 (cited on page 18).

[15] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. 6th. Springer Publishing Company, Incorporated, 2018. ISBN: 978-3-662-56038-9 (cited on page 19).

[16] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. USA: Prentice-Hall, Inc., 1982. ISBN: 0131524623 (cited on pages 17, 18).

[17] Claus Scheiderer. “Spectrahedral Shadows”. In: *SIAM J. Appl. Algebra Geom.* 2.1 (2018), pages 26–44 (cited on page 178).

[18] Naum Z Shor. “Utilization of the operation of space dilatation in the minimization of convex functions”. In: *Cybernetics* 6.1 (1972), pages 7–15 (cited on page 19).

[19] Naum Z Shor. “Cut-off method with space extension in convex programming problems”. In: *Cybernetics* 13.1 (1977), pages 94–96 (cited on page 19).

[20] Daniel A. Spielman and Shang-Hua Teng. “Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time”. In: *Journal of the ACM* 51.3 (2004), pages 385–463 (cited on page 18).

[21] Wikipedia. *Klee–Minty cube*. URL: `https://en.wikipedia.org/wiki/Klee%E2%80%93Minty_cube` (cited on page 18).

[22] Wikipedia. *Random vectors on a sphere*. URL: `https://en.wikipedia.org/wiki/N-sphere` (cited on page 171).

[23] Wikipedia. *Weyl’s inequality*. URL: `https://en.wikipedia.org/wiki/Weyl%27s_inequality` (cited on pages 157, 159).

[24] DB Yudin and Arkadi S Nemirovskii. “Informational complexity and efficient methods for the solution of convex extremal problems”. In: *Matekon* 13.2 (1976). In Russian, pages 22–45 (cited on page 19).

Index

D

E

F

G

H

I

K

L

T

V

W