Exercises in Convex Optimization
Lecture 7

Michel Baes, Patrick Cheridito

November 8, 2018

Please note that the third and the fourth exercises require Matlab. The third "exercise" is merely a set of instructions to install an efficient solver and to check that it works correctly.

Convex modeling We consider the problem of minimizing the convex function $f_0 : \mathbb{R}^n \mapsto \mathbb{R}$ over the convex hull of the union of some convex sets, $\text{conv}(\cup_{i=1}^q Ci)$. These sets are described via convex inequalities,

$$C_i = \{x | f_{ij}(x) \leq 0; j = 1, ..., k_i\}, \tag{1}$$

where $f_{ij} : \mathbb{R}^n \mapsto \mathbb{R}$ are convex. Our goal is to formulate this problem as a convex optimization problem.

The naive approach is to introduce variables $x_1, ..., x_q \in \mathbb{R}^n$, with $x_i \in C_i, \theta \in \mathbb{R}^q$ with $\theta \geq 0$, $\mathbf{1}^T\theta = 1$, and a variable $x \in \mathbb{R}^n$, with $x = \theta_1 x_1 + ... + \theta_q x_q$. This equality constraint is not affine in the variables, so this approach does not yield a convex problem. A more sophisticated formulation is given by

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & s_i f_{ij}(z_i/s_i) \leq 0; i = 1, \ldots, q; j = 1, \ldots, k_i \\ & 1^T s = 1; s \geq 0 \\ & x = z_1 + \cdots + z_q, \end{array}$$

with variables $z_1, \ldots, z_q \in \mathbb{R}^n, x \in \mathbb{R}^n$, and $s_1, \ldots, s_q \in \mathbb{R}$. (When $s_i = 0$, we take $s_i f_{ij}(z_i/s_i)$ to be 0 if $z_i = 0$ and ∞ if $z_i \neq 0$.) Explain why this problem is convex, and equivalent to the original problem.

Note: the function $(s, x) \mapsto sf(z/s)$ for $s > 0$, $z \in \text{dom} f$; $(0, z) \mapsto \chi_0$ is called the *perspective function* of f.

`Lecture_8/MinimizationConvexFunctionOverUnionConvexSets`

Convexity of some spread measures **[Boyd, Vandenberghe]** Given a set of real numbers $x_1, \ldots, x_n$, one can be interested in measuring how "different" these numbers are. Various notions of *spread* exist to measure this difference.

Standard spread. This measure is also called the *variation* of x:

$$f_1(x) := \max_{1\le i\le n} x_i - \min_{1\le i\le n} x_i.$$

Standard deviation. We consider a probability distribution on $\{x_1, \ldots, x_n\}$, where each number has a probability $1/n$. The spread measure here is the standard deviation of this distribution:

$$f_2(x) := \sqrt{\sum_{i=1}^n \frac{1}{n}\left(x_i - \sum_{j=1}^n \frac{x_j}{n}\right)^2} = \sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2 - \left(\sum_{j=1}^n \frac{x_j}{n}\right)^2}.$$

Deviation from the median. The median of x, denoted $\mathrm{med}(x)$, is defined as follows. If $n = 2k-1$, then $\mathrm{med}(x)$ is the kth largest component of x. If $n = 2k$, $\mathrm{med}(x)$ is the average of the kth and the $(k+1)$th component of x. The deviation from $\mathrm{med}(x)$ is defined as:

$$f_3(x) := \frac{1}{n}\sum_{i=1}^n |x_i - \mathrm{med}(x)|.$$

Prove that these three spread measures are convex.

`Lecture_8/Spread`

Introduction to Sedumi This exercise is merely a description of the software *Sedumi*.

Sedumi (SElf-DUal MInimization) is a free optimization software that solves optimization problems of the type:

$$\begin{array}{ll} \min & \langle c, x\rangle \\ \text{s.t.} & Ax = b \\ & x \in K \end{array} \tag{2}$$

where the cone K is any product of the following elementary cones:

$$\mathbb{R}^n_+,\ \mathbb{S}^N_+ \ \text{ and } \ \mathbb{L}^m_+ := \{(t,x) \in \mathbb{R}\times\mathbb{R}^{m-1} \mid t \ge \|x\|_2\}.$$

In order to ease your life, we have reproduced below some relevant parts of the help of Sedumi.

◇ **Installing Sedumi**

You can download Sedumi for free from `http://sedumi.ie.lehigh.edu`. After having unzipped it in any folder of your choice, you must set its path on Matlab (`File -> Set path`, click "Add with Subfolders", browse to your `SeDuMi_1_3` directory and click on "Save"). Finally, in Matlab go to the directory of Sedumi ("current folder") (and type `install_sedumi` in the command window). After some C compilation messages you should hopefully read "SeDuMi has been succesfully installed".

In order to check that everything is fine, type in the command line `x=sedumi([1,2],1,[1,1])`, making Sedumi solve the problem $\min\{x_1 + x_2 \mid x_1 + 2x_2 = 1, x \ge 0\}$. If everything is ok, it should find the (sparse) solution vector $x = [0\ ;\ 0.5]$.

◇ **Short help for Sedumi's Syntax**

In order to let Sedumi solve our optimization problem (2), we need to know how to encode it in the Sedumi's "language". Below are the most relevant parts of Sedumi's help. More details are given if you type `help sedumi`. Note that Sedumi is **extremely strict** with respect to the ordering of the variables!

```
[x,y,info] = sedumi(A,b,c,K,pars)

SEDUMI:  Self-Dual-Minimization / Optimization over self-dual homogeneous cones.

>  X = SEDUMI(A,b,c) yields an optimal solution to the linear program
     MINIMIZE c'*x SUCH THAT A*x = b, x >= 0
     x is a vector of decision variables.
     If size(A,2)==length(b), then it solves the linear program
     MINIMIZE c'*x SUCH THAT A'*x = b, x >= 0

>  [X,Y,INFO] = SEDUMI(A,b,c) also yields a vector of dual multipliers Y and a
     structure INFO, with the fields INFO.pinf, INFO.dinf and INFO.numerr.

>  [X,Y,INFO] = SEDUMI(A,b,c,K) instead of the constraint "x>=0", this
     restricts x to a self-dual homogeneous cone that you describe in the
     structure K. Up to 5 fields can be used, called K.f, K.l, K.q, K.r and
     K.s, for Free, Linear, Quadratic, Rotated-Quadratic and Semi-Definite.
     In addition, there are fields K.xcomplex, K.scomplex and K.ycomplex
     for complex-variables.

     (1) K.f is the number of FREE, i.e. UNRESTRICTED primal components.
         The dual components are restricted to be zero. E.g. if K.f = 2
         then x(1:2) is unrestricted, and z(1:2)=0.
         These are ALWAYS the first components in x.

     (2) K.l is the number of NON-NEGATIVE components. E.g. if K.f=2, K.l=8
         then x(3:10) >=0.

     (3) K.q lists the dimensions of LORENTZ (quadratic, second-order cone)
         constraints. E.g. if K.l=10 and K.q = [3 7] then
             x(11) >= norm(x(12:13)),
             x(14) >= norm(x(15:20)).
         These components ALWAYS immediately follow the K.l nonnegative ones.
         If the entries in A and/or c are COMPLEX, then the x-components in
         "norm(x(#,#))" take complex-values, whenever that is beneficial.
         Use K.ycomplex to impose constraints on the imaginary part of A*x.

     (4) K.r lists the dimensions of ROTATED-LORENTZ constraints.
         E.g. if K.l=10, K.q = [3 7] and K.r = [4 6], then
             2*x(21)x(22) >= norm(x(23:24))^2,
```

```
        2*x(25)x(26) >= norm(x(27:30))^2.
    These components ALWAYS immediately follow the K.q ones.
    Just as for the K.q-variables, the variables in "norm(x(#,#))" are
    allowed to be complex, if you provide complex data. Use K.ycomplex
    to impose constraints on the imaginary part of A*x.

(5) K.s lists the dimensions of POSITIVE SEMI-DEFINITE (PSD) constraints
    E.g. if K.l=10, K.q = [3 7] and K.s = [4 3], then
        mat( x(21:36),4 ) is PSD,
        mat( x(37:45),3 ) is PSD.
    These components are ALWAYS the last entries in x.

The dual multipliers y have analogous meaning as in the "x>=0" case,
except that instead of "c-A'*y>=0" resp. "-A'*y>=0", one should read that
c-A'*y resp. -A'*y are in the cone that is described by K.l, K.q and K.s.
In the above example, if z = c-A'*y and mat(z(21:36),4) is not symmetric/
Hermitian, then positive semi-definiteness reflects the symmetric/
Hermitian parts, i.e. Z + Z' is PSD.
```

◇ **Resolution of a Convex Problem with Sedumi: The Smallest Ball Problem**

We have a set of m points $a^{(i)} \in \mathbb{R}^n$, and we wish to find the smallest Euclidean ball $B_m(c,r)$ that *contains* them all. We will denote by A the points-matrix $A := (a^{(1)}, \ldots, a^{(m)})$.

The minimal ball problem can be formulated as follows:

$$\min\{r \mid \|a^{(i)} - c\|_2 \leq r, \quad \forall i = 1, \ldots, m\},$$

where $c \in \mathbb{R}^n$ denotes the center of the ball and $r > 0$ the radius.

In order to use Sedumi, we have to reformulate the problem to give it the form of (2) as follows:

$$\begin{array}{ll} \min & r \\ \text{s.t.} & \|a^{(i)} - c\|_2 \leq r, \forall i = 1, \ldots, m \\ & c \in \mathbb{R}^n, \ r \geq 0 \end{array} \iff$$

$$\begin{array}{lll} \min & r & \\ \text{s.t.} & y^{(i)} = a^{(i)} - c & \forall i = 1, \ldots, m \\ & r = t^{(i)} & \forall i = 1, \ldots, m \\ & c \in \mathbb{R}^n, \ r \geq 0 & \\ & (t^{(i)}, y^{(i)}) \in \mathbb{L}_+^{1+n} & \forall i = 1, \ldots, m \end{array}$$

Remark that now the optimization variable is

$$x = (c, r, (t^{(1)}, y^{(1)}), \ldots, (t^{(m)}, y^{(m)})^T \in \mathbb{R}^{n+1+m(n+1)}.$$

This (apparently more complicated) problem now has exactly the form of (2). Having again a look at Sedumi's syntax, here is a Matlab code that solves the (reformulated) smallest ball problem above.

```
function [center,r] = smallest_ball(A)
% INPUT       A: an n x m matrix containing the coordinates
%                of the m input points on its columns
% OUTPUT center: an n-dimensional vector, representing the
%                coordinates of the solution ball's center
%             r: the radius of the solution ball
% We use SeDuMi to deal with this problem

[n,m] = size(A);

K.f = n;
K.l = 1;
K.q = (n+1) * ones(1,m);

A_sedumi = spalloc(m*(n+1), (m+1)*(n+1), 2*m*(n+1));
for i = 1:m,
    for j = 1:n,
        A_sedumi(n*(i-1)+j,j)=1;
        A_sedumi(n*(i-1)+j,i*(n+1)+j+1)=1;
    end
    A_sedumi(m*n+i,n+1) = -1;
    A_sedumi(m*n+i,1+i*(n+1)) = 1;
end
b_sedumi = [reshape(A,n*m,1);zeros(m,1)];
c_sedumi = spalloc((m+1)*(n+1),1,1);
c_sedumi(n+1,1) = 1;

x = sedumi(A_sedumi,b_sedumi,c_sedumi,K);

center = x(1:n,1);
r = x(n+1,1);
% Note that (t^(i),y^(i)) = x(i*(n+1)+1:(i+1)*(n+1) , 1)
```

You can download this code from the course website.

Test the code for the points $\begin{pmatrix} 3 \\ -3 \end{pmatrix}, \begin{pmatrix} 5 \\ -5 \end{pmatrix}, \begin{pmatrix} 1 \\ 3 \end{pmatrix}$. You can visualize the solution by using the function `elliplot.m`, also available on the course website.

Test the code with 100 random points (use `randn` to generate them), then with 5000 random points.

`Lecture_8/Sedumi1Intro`

Bounding portfolio risk with incomplete covariance information We have to manage a portfolio of n assets. The quantity x_i denotes the amount of asset i we currently own. This quantity can be negative in case of short-selling. We would like to have more information on the risk of our positions. In the classical Markowitz setting, the risk of our portfolio is described by the quantity $x^T \Sigma x$, where Σ is the $n \times n$ correlation matrix between the different assets.

Suppose that we know the matrix Σ only (very) partially. We know its diagonal elements Σ_{ii}, which are the squares of the price volatilities of the assets. The off-diagonal entries of Σ are only known by their sign (or, in some cases, this sign is not even known). Intuitively, if $\Sigma_{ij} > 0$, then the price of assets i and j will tend to rise and fall together, like the price of oil and the price of an Exxon share.

Write a piece of Matlab code that computes the worst-case variance of our portfolio return and the corresponding matrix Σ. Compare this risk to the one where the matrix Σ is diagonal.

Test your code with the following instance:

$$x = \begin{pmatrix} 0.1 \\ 0.2 \\ -0.05 \\ 0.1 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 0.2 & + & + & \\ + & 0.1 & - & - \\ + & - & 0.3 & + \\ & - & + & 0.1 \end{pmatrix}.$$

Empty spots in Σ mean that the sign of the corresponding coefficient is unspecified.

`Lecture_8/SedumiIncompleteCovariance`