EE290N L. El Ghaoui 10/8/99

Homework Assignment #5 Solutions

1. *Dual of SOCP.*

   - Introducing the new variables, we have the problem

     $$\begin{array}{ll} \text{minimize} & e^T x \\ \text{subject to} & \|y_i\| \leq t_i \qquad i = 1, \ldots, N \\ & y = Ax + b \\ & t = C^T x + d, \end{array}$$

     for appropriate A, b, C, d.

   - The Lagrangian is given by

     $$\begin{array}{rcl} L(x, y, t, \lambda, h, \nu) & = & e^T x + \sum_{i=1}^N \lambda_i (\|y_i\| - t_i) + h^T (y - Ax - b) \\ & & + \nu^T (t - C^T x - d) \end{array}$$

     where $\lambda \succeq 0$ penalizes the violation of inequality constraints, and h, ν correspond to equality constraints.

     By minimizing over x and t, it is clear that the Lagrangian is unbounded below if

     $$e \neq A^T h + C^T \nu \text{ or } \nu \neq \lambda.$$

     Otherwise, we have

     $$L(x, y, t, \lambda, h, \nu) = -\lambda^T d - h^T b + h^T y + \sum_{i=1}^N \lambda_i \|y_i\|.$$

     The dual function is obtained by minimizing the above over y. If $\lambda_i = 0$ we have to choose $h_i = 0$ too. For $\lambda_i > 0$, we are led to solve the problem

     $$\text{minimize } \tilde{h}_i^T y_i + \|y_i\|_2,$$

     where $\tilde{h}_i = h_i / \lambda_i$. The above involves the expression of the conjuguate of the Euclidean norm function, so we have

     $$\inf_y \ \tilde{h}_i^T y_i + \|y_i\|_2 = \begin{cases} 0 & \text{if } \|\tilde{h}_i\|_2 \leq 1, \\ -\infty & \text{else.} \end{cases}$$

     (See reader, page 108). We obtain the Lagrange dual function

     $$g(\lambda, h, \nu) = \begin{cases} -\lambda^T d + h^T b & \text{if } \nu = \lambda \succeq 0, \quad e = A^T h + C^T \nu, \quad \|\tilde{h}\|_2 \leq 1. \\ -\infty & \text{else.} \end{cases}$$

- The Lagrange dual problem is, making all necessary equality constraints explicit,

$$\begin{array}{ll}\text{minimize} & \lambda^T d + h^T b \\ \text{subject to} & \lambda \succeq 0 \\ & e = A^T h + C^T \nu \\ & \left\| \frac{h_i}{\lambda_i} \right\| \leq 1, \quad i = 1, \ldots, N\end{array}$$

which in turn can be simplified to

$$\begin{array}{ll}\text{minimize} & \lambda^T d + h^T b \\ \text{subject to} & e = A^T h + C^T \nu \\ & \|h_i\| \leq \lambda_i, \quad i = 1, \ldots, N\end{array}$$

an SOCP with equality constraints. Note that in this formulation if we take $\lambda_i = 0$ we will necessarily need $h_i = 0$, so that the assumption made in last item is automatically taken care of.

2. *Portfolio optimization under incomplete covariance information.*

(a) First we write our problem as stated. We want to solve the problem

$$\begin{array}{ll}\text{maximize} & \left(x^T \Sigma x\right)^{1/2} \\ \text{subject to} & \Sigma \in \mathcal{E}\end{array}$$

We note that, since $\Sigma \succeq 0$, maximizing the square root of $x^T \Sigma x$ is equivalent to maximizing $x^T \Sigma x$ itself, that is, the problem is equivalent to

$$\begin{array}{ll}\text{maximize} & x^T \Sigma x \\ \text{subject to} & \Sigma \in \mathcal{E}\end{array}$$

The objective function is linear (hence convex) in Σ. We now turn our attention to the set $\mathcal{E}$. First of all, we need $\Sigma \succeq 0$, which defines a convex set. For each element of Σ, we have either a linear inequality constraint (a half-space), a equality constraint or nothing. All these constraints define a convex set, from which we conclude that $\mathcal{E}$ is a convex set. Therefore, the problem just posed is a convex optimization problem. In standard form, we can write it as

$$\begin{array}{ll}\text{minimize} & -x^T \Sigma x \\ \text{subject to} & \Sigma \succeq 0 \\ & \Sigma_{ij} \leq 0, \ \text{for } (i,j) \in I_+ \\ & \Sigma_{ij} \geq 0, \ \text{for } (i,j) \in I_- \\ & \Sigma_{ij} = \Sigma^0_{ij} \ \text{for } (i,j) \in I_0\end{array} \tag{1}$$

(b) We can solve the problem with the following `matlab` code (generated using `lmitool`):

```
function [Sigma, infostr] = maxrisk(Sign, Sigma0, x)
% function [Sigma, infostr] = maxrisk(Sign, Sigma0, x)
% created by TKLMITOOL-2.0 on 12-Oct-1999
% Sign is a matrix that is zero for known elements,
% infinite for unknown elements, and +/-1 otherwise.
% Sigma0 is the matrix with the known elements where Sigma
% is known, and arbitrary values otherwise.

% choose an optimization method
% TO CHANGE 'method', USE TKLMITOOL
method = 'sp' ;

% optimization parameters
Mbnd=1e4 ; abstol=1e-10 ; nu=10 ; maxiters=100 ; reltol=1e-10;

% Format parameters
sym_chck=1 ; lin_chck=1 ; eqtol=1e-8 ; eqtol2=1e-10 ;

options = [Mbnd abstol nu maxiters reltol sym_chck lin_chck eqtol eqtol2] ;

% stack data into one vector
xdata = stack (Sign, Sigma0, x) ;

% invoke lmisolver
[xvar, infostr] = lmisolver ('maxrisk_eval', xdata, method, options) ;

% unstack result
[Sigma] = unstack (xvar) ;

function [LME, LMI, OBJ] = maxrisk_eval(xdata, xvar)
% function [LME, LMI, OBJ] = maxrisk_eval(xdata, xvar)
% maxrisk_eval : evaluation function for lmisolver.
% created by TKLMITOOL-2.0 on 12-Oct-1999
% see also maxrisk.m

[Sign, Sigma0, x] = unstack (xdata) ;

% PRELIMINARY COMPUTATION
n = length(x);
Iplus = find(Sign==1);
Iminus = find(Sign==-1);
I0 = find(Sign==0);

% INITIAL GUESS (need not be feasible)
if nargin == 1,
Sigmainit = eye(n);

LME = Sigmainit ;
return, end

% LME, LMI, and OBJECTIVE
[Sigma] = unstack (xvar) ;

% Compute LMEs
% make sure Sigma is symmetric
LME1 = Sigma'-Sigma;
% make sure that whenever the Sign matrix is zero,
% Sigma coincides with Sigma0
LME2 = diag(Sigma-Sigma0);
%LME2 = Sigma(I0)-Sigma0(I0);
% combine both equality constraints
LME = stack(LME1,LME2);

% Compute LMIs
LMI = [];
% make sure Sigma is positive at appropriate places
for i = 1:length(Iplus),
  LMI = stack(LMI,Sigma(Iplus(i)));
end
% make sure Sigma is negative at appropriate places
for i = 1:length(Iminus),
  LMI = stack(LMI,-Sigma(Iminus(i)));
end
% finally enforce Sigma to be PSD
LMI = stack(LMI,Sigma);

% Compute Linear Objective
OBJ = -x'*Sigma*x;
```

Running this script we get the optimal value to be 0.0152, that is, we get $\sigma_{\mathrm{wc}} =$

0.1232. The associated Σ matrix is given by

$$\Sigma = \begin{bmatrix} 0.2000 & 0.0817 & 0.0000 & 0.1064 \\ 0.0817 & 0.1000 & -0.1333 & -0.0000 \\ 0.0000 & -0.1333 & 0.3000 & 0.0644 \\ 0.1064 & -0.0000 & 0.0644 & 0.1000 \end{bmatrix}.$$

As we can see, all inequality constraints are satisfied, and a call to Matlab's `eig` function will show that the matrix is positive semidefinite.

3. Let Σ^0 be a (symmetric) matrix whose elements are Σ_{ij} when Σ_{ij} is known (that is, when $(i,j) \in I_0$), and zero otherwise. Let I_∞ be the set of indices (i,j) where the elements Σ_{ij} are completely unknown.

   The primal problem is (1). We form the Lagrangian

$$L = -x^T \Sigma x - \mathbf{Tr}\, \Sigma X + \sum_{1 \le i \le j \le n} P_{ij}(\Sigma_{ij} - \Sigma^0_{ij}),$$

   where the multiplier $X \succeq 0$ corresponds to the positive semidefiniteness constraint on Σ, and the multiplier matrix P is constrained to belong to the polytope

$$\mathcal{P} := \left\{ P = P^T \mid P_{ij} \le 0 \text{ if } (i,j) \in I_+, \;\; P_{ij} \ge 0 \text{ if } (i,j) \in I_-, \;\; P_{ij} = 0 \text{ if } (i,j) \in I_\infty \right\}.$$

   (The choice of the sign constraints is such that a violation of the constraints of the primal problem should result in an increase in the Lagrangian.)

   Minimizing the Lagrangian over Σ (now a free parameter) yields $-\infty$, except when

$$xx^T + X = P.$$

   In this case, the value of the Lagrangian is

$$L(\Sigma, X, P) = -\sum_{(i,j) \in I_0} P_{ij} \Sigma^0_{ij} = -\mathbf{Tr}\, P \Sigma^0.$$

   So, the dual function is:

$$g(X,P) = \begin{cases} -\mathbf{Tr}\, P\Sigma^0 & \text{if } xx^T + X = P, \;\; X \succeq 0, \;\; P \in \mathcal{P}, \\ -\infty & \text{else.} \end{cases}$$

   After eliminating X to make the inequality constraint explicit, we obtain the dual problem

$$\text{maximize } -\mathbf{Tr}\, P\Sigma^0 \text{ subject to } \;\; P \succeq xx^T, \;\; P \in \mathcal{P}. \tag{2}$$

   We note that, in the case all diagonal elements of Σ are known, then both the primal and dual problem are strictly feasible. Hence, both primal and dual problems are attained, and the duality gap is zero at the optimum. In this case, the optimal value of the above problem is $-\sigma_{\mathrm{wc}}(x)^2$.

4. Let $\mathcal{X}$ denote the set of feasible investments, which in our case is the convex set

$$\mathcal{X} = \left\{ x \mid \overline{p}^T x \geq r, \ \ \sum_{i=1}^n x_i \leq S, \ \ x_i \geq 0 \right\}.$$

(We replaced the equality constraint by an inequality, to be more general.)
In view of the dual formulation (2), the problem expresses as

$$\min_{x \in \mathcal{X}} \ \sigma(x)^2 = \min_{P,x} \ \mathbf{Tr}\, P\Sigma^0 \ \text{subject to} \quad P \succeq xx^T, \ \ P \in \mathcal{P}, \ \ x \in \mathcal{X}.$$

The above problem is convex in both P and x, since the constraint

$$P \succeq xx^T$$

is equivalent to the linear matrix inequality in P, x

$$\left[\begin{array}{cc} P & x \\ x^T & 1 \end{array} \right] \succeq 0.$$

In fact, the problem of minimizing the worst-case risk can be written as a SDP:

$$\text{minimize } \mathbf{Tr}\, P\Sigma^0 \ \text{subject to} \quad \left[\begin{array}{cc} P & x \\ x^T & 1 \end{array} \right] \succeq 0, \ \ P \in \mathcal{P}, \ \ x \in \mathcal{X}.$$

The following `matlab` code solves the problem.

```
function [x, P, infostr] = minmaxrisk(S, p, r, Sigma0, Sign)
% function [x, P, infostr] = minmaxrisk(S, p, r, Sigma0, Sign)
% created by TKLMITOOL-2.0 on 12-Oct-1999

% choose an optimization method
% TO CHANGE 'method', USE TKLMITOOL
method = 'sdp' ;

options = setoptions(method);

% stack data into one vector
xdata = stack (S, p, r, Sigma0, Sign) ;

% invoke lmisolver
[xvar, infostr] = lmisolver ('minmaxrisk_eval', xdata, method, options) ;

% unstack result
[x, P] = unstack (xvar) ;

function [LHE, LHI, OBJ] = minmaxrisk_eval(xdata, xvar)
% function [LHE, LHI, OBJ] = minmaxrisk_eval(xdata, xvar)
% minmaxrisk_eval : evaluation function for lmisolver.
% created by TKLMITOOL-2.0 on 12-Oct-1999
% see also minmaxrisk.m

[S, p, r, Sigma0, Sign] = unstack (xdata) ;

% PRELIMINARY COMPUTATION
n = length(p);
Iplus = find(Sign==1);
Iminus = find(Sign==-1);
I0 = find(Sign==0);
Iinf = find(~finite(Sign));

% INITIAL GUESS (need not be feasible)
if nargin == 1,
xinit = zeros(n,1);
Pinit = zeros(n,n);

LHE = stack(xinit,Pinit);
return, end
```

```
% LHE, LHI, and OBJECTIVE
[x, P] = unstack (xvar) ;

% Compute LHEs
LHE = [];
% make sure P is symmetric
LHE = P-P';
% make sure P is zero whenever Sigma is unknown
LHE = stack(LHE,P(Iinf));

% Compute LHIs
% make sure total wealth is not exceeded
LHI = S-sum(x);
% constraint on minimum return
LHI = stack(LHI,p'*x-r);
% we do not allow shorting, so x is positive
for i = 1:n,
  LHI = stack(LHI,x(i));
end
% constraint that couples x and P
LHI = stack(LHI,[P x; x' 1]);
% make sure P is negative whenever Sigma is positive
for i =1:length(Iplus),
  LHI = stack(LHI,-P(Iplus(i)));
end
% make sure P is positive whenever Sigma is negative
for i = 1:length(Iminus),
  LHI = stack(LHI,P(Iminus(i)));
end

% Compute Linear Objective
OBJ = trace(P*Sigma0);
```

The solution obtained (using the solver SDPpack) is

$$P = \begin{bmatrix} 0.0000 & -0.0000 & -0.0000 & 0.0000 \\ -0.0000 & 532.5444 & 0.0000 & 355.0296 \\ -0.0000 & 0.0000 & 0.0000 & -0.0000 \\ 0.0000 & 355.0296 & -0.0000 & 236.6864 \end{bmatrix}, \quad x = \begin{bmatrix} 0.0000 \\ 23.0769 \\ 0.0000 \\ 15.3846 \end{bmatrix}.$$

Note that the optimization does not recommend to invest all assets, that is, the sum $x_1 + \ldots + x_n$ is less than the total wealth $S = 1000$.