CVX101 Homework 3 solutions

4.1 Consider the optimization problem

$$\begin{array}{ll} \text{minimize} & f_0(x_1, x_2) \\ \text{subject to} & 2x_1 + x_2 \geq 1 \\ & x_1 + 3x_2 \geq 1 \\ & x_1 \geq 0, \quad x_2 \geq 0. \end{array}$$

Make a sketch of the feasible set. For each of the following objective functions, give the optimal set and the optimal value.

(a) $f_0(x_1, x_2) = x_1 + x_2$.

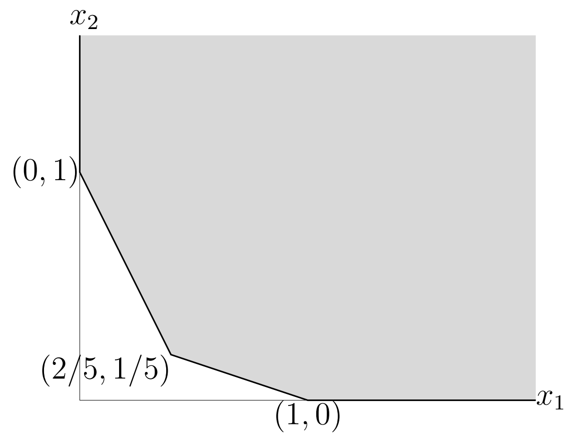
(b) $f_0(x_1, x_2) = -x_1 - x_2$.

(c) $f_0(x_1, x_2) = x_1$.

(d) $f_0(x_1, x_2) = \max\{x_1, x_2\}$.

(e) $f_0(x_1, x_2) = x_1^2 + 9x_2^2$.

Solution. The feasible set is shown in the figure.



(a) $x^\star = (2/5, 1/5)$.

(b) Unbounded below.

(c) $X_{\text{opt}} = \{(0, x_2) \mid x_2 \geq 1\}$.

(d) $x^\star = (1/3, 1/3)$.

(e) $x^\star = (1/2, 1/6)$. This is optimal because it satisfies $2x_1+x_2 = 7/6 > 1$, $x_1+3x_2 = 1$, and

$$\nabla f_0(x^\star) = (1, 3)$$

is perpendicular to the line $x_1 + 3x_2 = 1$.

A3.2 *'Hello World' in CVX.* Use CVX to verify the optimal values you obtained (analytically) for exercise 4.1 in *Convex Optimization.*

Solution.

(a) $p^\star = 0.6$

(b) $p^\star = -\infty$

(c) $p^\star = 0$

(d) $p^\star = \frac{1}{3}$

(e) $p^\star = \frac{1}{2}$

```
%exercise 4.1 using CVX

%set up a vector to store optimal values of problems
optimal_values=zeros(5,1);

%part a
cvx_begin
variable x(2)
minimize(x(1)+x(2))
2*x(1)+x(2) >= 0;
x(1)+3*x(2) >= 1;
x >= 0;
cvx_end

optimal_values(1)=cvx_optval;

%part b
cvx_begin
variable x(2)
minimize(-sum(x))
2*x(1)+x(2) >= 0;
x(1)+3*x(2) >= 1;
x >= 0;
cvx_end
optimal_values(2)=cvx_optval;

%part c
cvx_begin
variable x(2)
minimize(x(1))
2*x(1)+x(2) >= 0;
```

```
x(1)+3*x(2) >= 1;
x >= 0;
cvx_end
optimal_values(3)=cvx_optval;

%part d
cvx_begin
variable x(2)
minimize(max(x))
2*x(1)+x(2) >= 0;
x(1)+3*x(2) >= 1;
x >= 0;
cvx_end
optimal_values(4)=cvx_optval;

%part e
cvx_begin
variable x(2,1)
minimize( square(x(1))+ 9*square(x(2)) )
2*x(1)+x(2) >= 0;
x(1)+3*x(2) >= 1;
x >= 0;
cvx_end
optimal_values(5)=cvx_optval;
```

4.15 *Relaxation of Boolean LP.* In a *Boolean linear program*, the variable x is constrained to have components equal to zero or one:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \preceq b \\ & x_i \in \{0,1\}, \quad i = 1, \ldots, n. \end{array} \tag{1}$$

In general, such problems are very difficult to solve, even though the feasible set is finite (containing at most 2^n points).

In a general method called *relaxation*, the constraint that x_i be zero or one is replaced with the linear inequalities $0 \leq x_i \leq 1$:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \preceq b \\ & 0 \leq x_i \leq 1, \quad i = 1, \ldots, n. \end{array} \tag{2}$$

We refer to this problem as the *LP relaxation* of the Boolean LP (1). The LP relaxation is far easier to solve than the original Boolean LP.

(a) Show that the optimal value of the LP relaxation (2) is a lower bound on the optimal value of the Boolean LP (1). What can you say about the Boolean LP if the LP relaxation is infeasible?

(b) It sometimes happens that the LP relaxation has a solution with $x_i \in \{0, 1\}$. What can you say in this case?

Solution.

(a) The feasible set of the relaxation includes the feasible set of the Boolean LP. It follows that the Boolean LP is infeasible if the relaxation is infeasible, and that the optimal value of the relaxation is less than or equal to the optimal value of the Boolean LP.

(b) The optimal solution of the relaxation is also optimal for the Boolean LP.

A3.18 *Heuristic suboptimal solution for Boolean LP.* This exercise builds on exercises 4.15 and 5.13 in *Convex Optimization*, which involve the Boolean LP

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \preceq b \\ & x_i \in \{0,1\}, \quad i = 1, \ldots, n, \end{array}$$

with optimal value $p^\star$. Let $x^{\rm rlx}$ be a solution of the LP relaxation

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \preceq b \\ & 0 \preceq x \preceq \mathbf{1}, \end{array}$$

so $L = c^T x^{\rm rlx}$ is a lower bound on $p^\star$. The relaxed solution $x^{\rm rlx}$ can also be used to guess a Boolean point $\hat{x}$, by rounding its entries, based on a threshold $t \in [0, 1]$:

$$\hat{x}_i = \left\{ \begin{array}{ll} 1 & x_i^{\rm rlx} \geq t \\ 0 & \text{otherwise,} \end{array} \right.$$

for $i = 1, \ldots, n$. Evidently $\hat{x}$ is Boolean (*i.e.*, has entries in $\{0, 1\}$). If it is feasible for the Boolean LP, *i.e.*, if $A\hat{x} \preceq b$, then it can be considered a guess at a good, if not optimal, point for the Boolean LP. Its objective value, $U = c^T \hat{x}$, is an upper bound on $p^\star$. If U and L are close, then $\hat{x}$ is nearly optimal; specifically, $\hat{x}$ cannot be more than $(U - L)$-suboptimal for the Boolean LP.

This rounding need not work; indeed, it can happen that for all threshold values, $\hat{x}$ is infeasible. But for some problem instances, it can work well.

Of course, there are many variations on this simple scheme for (possibly) constructing a feasible, good point from $x^{\rm rlx}$.

Finally, we get to the problem. Generate problem data using

```
rand('state',0);
n=100;
m=300;
A=rand(m,n);
b=A*ones(n,1)/2;
c=-rand(n,1);
```

You can think of x_i as a job we either accept or decline, and $-c_i$ as the (positive) revenue we generate if we accept job i. We can think of $Ax \preceq b$ as a set of limits on m resources. A_{ij}, which is positive, is the amount of resource i consumed if we accept job j; b_i, which is positive, is the amount of resource i available.

Find a solution of the relaxed LP and examine its entries. Note the associated lower bound L. Carry out threshold rounding for (say) 100 values of t, uniformly spaced over $[0,1]$. For each value of t, note the objective value $c^T\hat{x}$ and the maximum constraint violation $\max_i(A\hat{x}-b)_i$. Plot the objective value and the maximum violation versus t. Be sure to indicate on the plot the values of t for which $\hat{x}$ is feasible, and those for which it is not.

Find a value of t for which $\hat{x}$ is feasible, and gives minimum objective value, and note the associated upper bound U. Give the gap $U-L$ between the upper bound on $p^\star$ and the lower bound on $p^\star$. If you define vectors `obj` and `maxviol`, you can find the upper bound as `U=min(obj(find(maxviol<=0)))`.

Solution. The following Matlab code finds the solution

```
% generate data for boolean LP relaxation & heuristic
rand('state',0);
n=100;
m=300;
A=rand(m,n);
b=A*ones(n,1)/2;
c=-rand(n,1);

% solve LP relaxation
cvx_begin
   variable x(n)
   minimize (c'*x)
   subject to
        A*x <= b
        x>=0
        x<=1
cvx_end
xrlx = x;
L=cvx_optval;
```

```
% sweep over threshold & round
thres=0:0.01:1;
maxviol = zeros(length(thres),1);
obj = zeros(length(thres),1);
for i=1:length(thres)
   xhat = (xrlx>=thres(i));
   maxviol(i) = max(A*xhat-b);
   obj(i) = c'*xhat;
end

% find least upper bound and associated threshold
i_feas=find(maxviol<=0);
U=min(obj(i_feas))
%U=min(obj(find(maxviol <=0)))
t=min(i_feas);
min_thresh=thres(t)

% plot objective and max violation versus threshold
subplot(2,1,1)
plot(thres(1:t-1),maxviol(1:t-1),'r',thres(t:end), ...
maxviol(t:end),'b','linewidth',2);
xlabel('threshold');
ylabel('max violation');
subplot(2,1,2)
hold on; plot(thres,L*ones(size(thres)),'k','linewidth',2);
plot(thres(1:t-1),obj(1:t-1),'r',thres(t:end), ...
obj(t:end),'b','linewidth',2);
xlabel('threshold');
ylabel('objective');
```

The lower bound found from the relaxed LP is $L = -33.1672$. We find that the threshold value $t = 0.6006$ gives the best (smallest) objective value for feasible $\hat{x}$: $U = -32.4450$. The difference is 0.7222. So $\hat{x}$, with $t = 0.6006$, can be no more than 0.7222 suboptimal, *i.e.*, around 2.2% suboptimal.

In figure 1, the red lines indicate values for thresholding values which give infeasible $\hat{x}$, and the blue lines correspond to feasible $\hat{x}$. We see that the maximum violation decreases as the threshold is increased. This occurs because the constraint matrix A only has nonnegative entries. At a threshold of 0, all jobs are selected, which is an infeasible solution. As we increase the threshold, projects are removed in sequence (without adding new projects), which monotonically decreases the maximum violation. For a general boolean LP, the corresponding plots need not exhibit monotonic behavior.

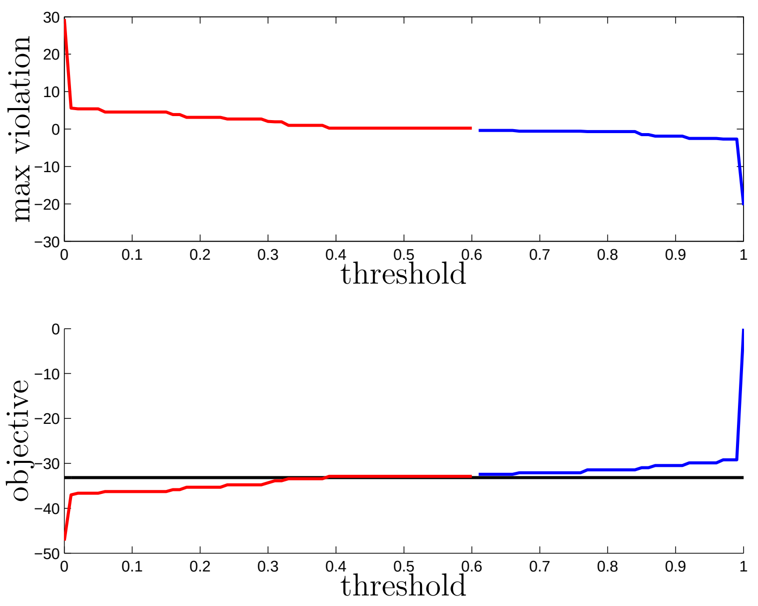


Figure 1 Plots of violation and objective vs threshold rule.

A13.3 *Simple portfolio optimization.* We consider a portfolio optimization problem as described on pages 155 and 185–186 of *Convex Optimization*, with data that can be found in the file `simple_portfolio_data.m`.

(a) Find minimum-risk portfolios with the same expected return as the uniform portfolio ($x = (1/n)\mathbf{1}$), with risk measured by portfolio return variance, and the following portfolio constraints (in addition to $\mathbf{1}^T x = 1$):

- No (additional) constraints.
- Long-only: $x \succeq 0$.
- Limit on total short position: $\mathbf{1}^T(x_-) \leq 0.5$, where $(x_-)_i = \max\{-x_i, 0\}$.

Compare the optimal risk in these portfolios with each other and the uniform portfolio.

(b) Plot the optimal risk-return trade-off curves for the long-only portfolio, and for total short-position limited to 0.5, in the same figure. Follow the style of figure 4.12 (top), with horizontal axis showing standard deviation of portfolio return, and vertical axis showing mean return.

Solution.

(a) We can express these as QPs:

- No (additional) constraints:

$$\begin{array}{ll} \text{minimize} & x^T \Sigma x \\ \text{subject to} & \mathbf{1}^T x = 1, \quad \bar{p}^T x = \bar{p}^T (1/n)\mathbf{1} \end{array}$$

- Long only

$$\begin{array}{ll} \text{minimize} & x^T\Sigma x \\ \text{subject to} & x \succeq 0, \quad \mathbf{1}^T x = 1 \\ & \overline{p}^T x = \overline{p}^T(1/n)\mathbf{1} \end{array}$$

- Limit on total short position:

$$\begin{array}{ll} \text{minimize} & x^T\Sigma x \\ \text{subject to} & \mathbf{1}^T x = 1, \quad \overline{p}^T x = \overline{p}^T(1/n)\mathbf{1} \\ & \mathbf{1}^T x_- \leq 0.5 \end{array}$$

Although the two portfolios have the same expected return, their risk profiles differ drastically. The standard deviations are:

- uniform: 8.7%
- long-only: 5.1%
- limit on total short position: 2.1%
- unconstrained: 1.9%.

Notice that as the size of the feasible set increases, the objective value improves.

(b) The optimal risk-return trade-off curves can be generated by scalarizing the bi-criterion objective, $\overline{p}^T x - \gamma x^T \Sigma x$, over a range of values for γ. For example, the optimal curve for the long-only portfolio is generated by solving the following family of QPs:

$$\begin{array}{ll} \text{maximize} & \overline{p}^T x - \gamma x^T\Sigma x \\ \text{subject to} & x \succeq 0, \quad \mathbf{1}^T x = 1 \end{array}$$

For every level of return, there is a portfolio with limits on total short positions (red) that has greater expected return than the optimal long-only portfolio (blue).

The following MATLAB script solves this problem:

```
simple_portfolio_data;
%% part i
%minimum-risk unconstrained portfolio
%with same expected return as uniform
%allocation
cvx_begin
cvx_quiet(true)
variable x_unconstrained(n)
minimize(quad_form(x_unconstrained,S))
subject to
    sum(x_unconstrained)==1;
    pbar'*x_unconstrained==x_unif'*pbar;
cvx_end
```

```
%% part ii
%minimum-risk long-only portfolio
%with same expected return as uniform
%allocation
cvx_begin
cvx_quiet(true)
variable x_long(n)
minimize(quad_form(x_long,S))
subject to
    x_long>=0;
    sum(x_long)==1;
    pbar'*x_long==x_unif'*pbar;
cvx_end
%% part iii
%minimum-risk constrained short portfolio
%with same expected return as uniform
%allocation
cvx_begin
cvx_quiet(true)
variable x_shortconstr(n)
minimize(quad_form(x_shortconstr,S))
subject to
    sum(pos(-x_shortconstr))<=0.5;
    sum(x_shortconstr)==1;
    pbar'*x_shortconstr==x_unif'*pbar;
cvx_end
%% Generate risk-return trade-off curves
sprintf('unconstrained sd: %0.3g\n', ...
sqrt(quad_form(x_unconstrained,S)))
sprintf('long only sd: %0.3g\n', sqrt(quad_form(x_long,S)))
sprintf('constrained short sd: %0.3g\n', ...
sqrt(quad_form(x_shortconstr,S)))
sprintf('x_unif sd: %0.3g\n', sqrt(quad_form(x_unif,S)))
novals=100;
r_long = [];
r_shortconstr = [];
sd_long = [];
sd_shortconstr = [];
muvals = logspace(-1,4,novals);

for i=1:novals
  mu  = muvals(i);
  %long only
  cvx_begin
  cvx_quiet(true)
  variable x(n)
  maximize(pbar'*x - mu*quad_form(x,S))
  subject to
    x>=0;
    sum(x)==1;
  cvx_end
  r_long = [r_long,  pbar'*x];
  sd_long = [sd_long,  sqrt(x'*S*x) ];
  %constrained short
  cvx_begin
  cvx_quiet(true)
  variables x(n)
  maximize(pbar'*x - mu*quad_form(x,S))
  subject to
    sum(x)==1;
    sum(pos(-x))<=.5;
  cvx_end
  r_shortconstr = [r_shortconstr, pbar'*x];
  sd_shortconstr = [sd_shortconstr, sqrt(x'*S*x)];
end;

plot(sd_long, r_long);
hold; plot(sd_shortconstr, r_shortconstr, 'r');
```