Solutions to hw4 homework on Convex Optimization https://web.stanford.edu/class/ee364a/homework.html

Andrei Keino

July 8, 2020

5.1

A simple example. Consider the optimization problem

$$\begin{array}{ll} \textit{minimize} & x^2 + 1 \\ \textit{subject to} & (x-2)(x-4) \leq 0 \end{array}$$

with variable $x \in R$

(a) Analysis of primal problem. Give the feasible set, the optimal value, and the optimal solution.

(b) Lagrangian and dual function. Plot the objective $x^2 + 1$ versus x. On the same plot, show the feasible set, optimal point and value, and plot the Lagrangian $L(x, \lambda)$ versus x for a few positive values of λ. Verify the lower bound property ($p^* \leq \inf_x L(x, \lambda)$ for $\lambda \geq 0$). Derive and sketch the Lagrange dual function g.

(c) Lagrange dual problem. State the dual problem, and verify that it is a concave maximization problem. Find the dual optimal value and dual optimal solution λ^*. Does strong duality hold?

(d) Sensitivity analysis. Let $p^*(u)$ denote the optimal value of the problem

$$\begin{array}{ll} \textit{minimize} & x^2 + 1 \\ \textit{subject to} & (x-2)(x-4) \leq u \end{array}$$

as a function of the parameter u. Plot $p^*(u)$. Verify that $dp^*(0)/du = -\lambda^*$.

Solution:

(a)

The feasible set is $x \in [2, 4]$. The optimal solution is $x^* = 2$, the optimal value is $p^* = 5$.

(b)

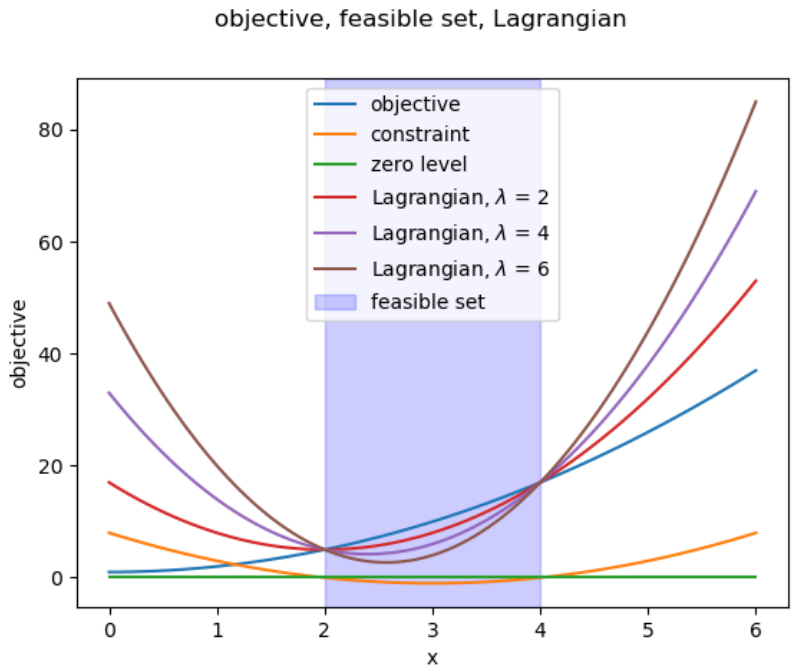

Figure 1: objective, feasible set, lagrangian for this problem.

It's easy to see from the Figure 1, that the Lagrangian values on the feasible set are less or equal than the objective values on the feasible set, i. e. $(p^* \geq \inf_x L(x, \lambda)\ for\ \lambda \geq 0)$.

(c)
The dual objective function for this problem can be found solving the constrained equation for the Lagrangian:

$$g(\lambda) = inf_x(x^2 + 1 + \lambda(x-2)(x-4))$$
$$subject\ to \qquad \lambda \geq 0$$

The Lagrangian reaches its minimum at the point $\tilde{x} = \frac{3\lambda}{\lambda+1}$. Then the dual objective itself is:

$$g(\lambda) = -\lambda + 10 - \frac{9}{(\lambda+1)}$$

The second derivative of the dual function is:

$$g^{''}(\lambda) = -18/(\lambda+1)^3$$

which is obviously less than zero for $\lambda \geq 0$, i.e. the dual function is concave for $\lambda \geq 0$.

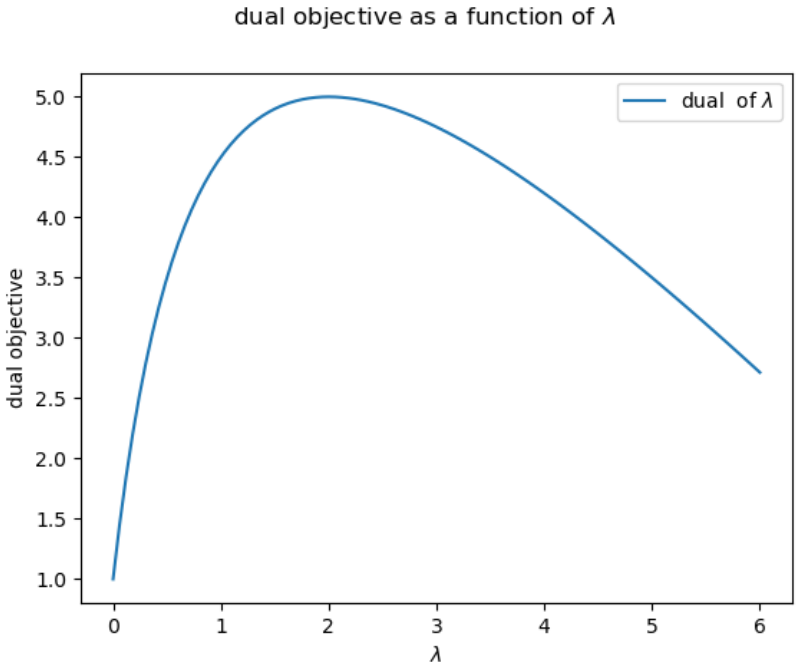

Figure 2: Dual objective for this problem.

The dual optimal value λ^* can be found solving the dual problem for the $g(\lambda)$:

$$\begin{aligned} &maximize && g(\lambda) \\ &subject\ to && \lambda \geq 0 \end{aligned}$$

or

$$\begin{aligned} &dg(\lambda)/d\lambda = 0 \\ &subject\ to && \lambda \geq 0 \end{aligned}$$

Solving the equation we found $\lambda^* = 2$; optimal value of g (i.e. $sup_\lambda\{g(\lambda)\}$) $g^* = 5$. We can see that $p^* = 5 = g^*$, i. e. strong duality holds.

(d)

Solving the constraints equation $(x-2)(x-4) = u$ we get: $x_{1,2} = 3 \pm \sqrt{1+u},\ u \geq 1$. Then

$$p^*(u) = \begin{cases} \text{not exists}, & \text{if } u < -1 \\ u - 6\sqrt{1+u} + 11, & \text{if } -1 \leq u \leq 8 \\ 1, & \text{if } u > 8 \end{cases} \tag{1}$$

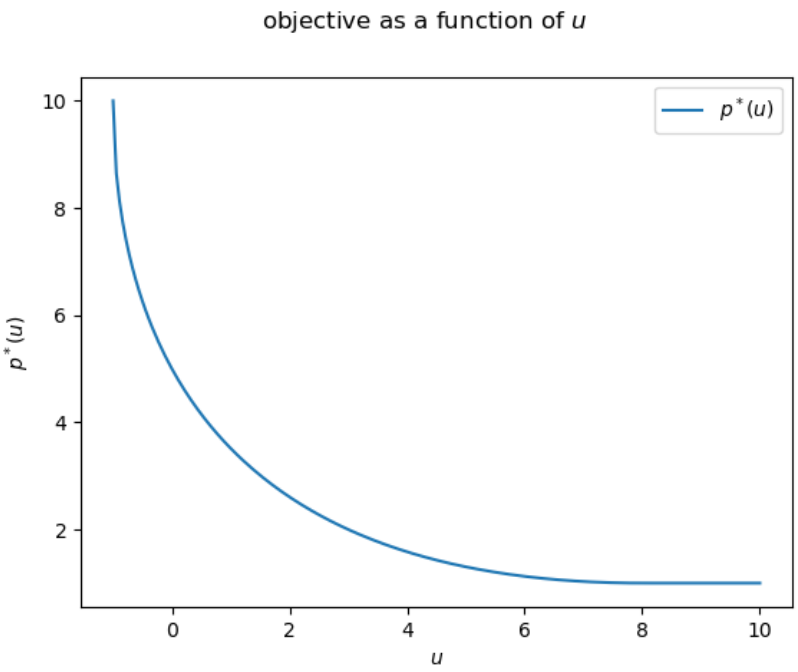

Figure 3: Graph of $p^*(u)$ for this problem.

And finally:

$$dp^*(u)/du = 1 - \frac{3}{\sqrt{1+u}}$$

Then $dp^*(0)/du = -2 = -\lambda^*$

5.3

Problems with one inequality constraint. Express the dual problem of

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } f(x) \leq 0 \end{aligned}$$

with $c \neq 0$ in terms of the conjugate f^*. Explain why the problem you give is convex. We do not assume f is convex.

Solution:
The dual problem of the task is:

$$\begin{aligned} &\text{maximize } \inf_x (c^T x + \lambda f(x)) \\ &\text{subject to } \lambda \geq 0 \end{aligned}$$

The definition of the conjugate function is:

$$f^*(y) = \sup_x (y^T x - f(x)) = -\inf_x (f(x) - y^T x)$$

i.e. the dual problem can be reformulated as:

$$\begin{aligned} &\text{maximize } F(c, \lambda) \\ &\text{subject to } \lambda \geq 0 \end{aligned}$$

where $F(c, \lambda) = -\lambda f^*(-c/\lambda)$, where f^* is the conjugate function of f, and it is always convex. $F(c, \lambda)$ is the concave function, as it is the negative perspective of the convex function f^*.

5.12

Analytic centering. Derive a dual problem for

$$\text{minimize } -\sum_{i=1}^{m} log(b_i - a_i^T x)$$

with domain $\{x \mid a_i^T x \leq b_i,\ i = 1, ..., m\}$. First introduce new variables y_i and equality constraints $y_i = b_i - a_i^T x$. (The solution of this problem is called the analytic center of the linear inequalities $a_i^T x \leq b_i,\ i = 1, ..., m\}$. Analytic centers have geometric applications and play an important role in barrier methods.

Solution:

This problem is equivalent to:

$$\begin{aligned} \text{minimize} \quad & -\sum_{i=1}^{m} log(y_i) \\ \text{subject to} \quad & y + Ax - b = 0 \end{aligned}$$

where the matrix A composed from the rows a_i^T, the i - th row of A is a_i^T
Then the Lagrangian is

$$L(x, y, \nu) = -\sum_{i=1}^{m} log(y_i) + \nu^T (y + Ax - b)$$

The dual function is

$$g(v) = \inf_{x,y}(-\sum_{i=1}^{m} log(y_i) + \nu^T (y + Ax - b))$$

The term $\nu^T Ax$ is unbounded below as $x \to \infty$, so

$$g(\nu) = \begin{cases} \sum_{i=1}^{m} log(y_i) + m - \nu^T b, & A\nu = 0,\ \nu_i \geq 0 \\ -\infty, & A\nu \neq 0 \end{cases}$$

and the dual problem is:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} log(\nu_i) + m - \nu^T b \\ \text{subject to} \quad & A\nu = 0 \end{aligned}$$

A3.3

Reformulating constraints in CVX*. Each of the following CVX code fragments describes a convex constraint on the scalar variables x, y, and z, but violates the CVX rule set, and so is invalid. Briely explain why each fragment is invalid. Then, rewrite each one in an equivalent form that conforms to the CVX rule set. In your reformulations, you can use linear equality and inequality constraints, and inequalities constructed using CVX functions. You can also introduce additional variables, or use LMIs. Be sure to explain (briefly) why your reformulation is equivalent to the original constraint, if it is not obvious. Check your reformulations by creating a small problem that includes these constraints, and solving it using CVX. Your test problem doesn't have to be feasible; it's enough to verify that CVX processes your constraints without error. Remark. This looks like a problem about 'how to use CVX software', or 'tricks for using

CVX'. But it really checks whether you understand the various composition rules, convex analysis, and constraint reformulation rules.

(a) norm([x + 2*y, x - y]) == 0
(b) square(square(x + y)) <= x - y
(c) 1/x + 1/y <= 1; x >= 0; y >= 0
(d) norm([max(x,1), max(y,2)]) <= 3*x + y
(e) x*y >= 1; x >= 0; y >= 0
(f) (x + y)^2/sqrt(y) <= x - y + 5
(g) x^3 + y^3 ¡= 1; x >= 0; y >= 0
(h) x + z ¡= 1 + sqrt(x*y - z^2); x >= 0; y >= 0

Solution:

(a) norm([x + 2*y, x - y]) == 0

This constraint is invalid because **Three types of constraints may be specified in disciplined convex programs:**

- An equality constraint, constructed using ==, where both sides are affine.
- A less-than inequality constraint, using <=, where the left side is convex and the right side is concave.
- A greater-than inequality constraint, using >=, where the left side is concave and the right side is convex.

`http://web.cvxr.com/cvx/doc/dcp.html#constraints`

Expression in (a) violates the first rule - both sides must be affine. As the given norm could be zero only if $(x, y) = (0, 0)$, the workaround:

The cvx test script:

```
subject to
x == 0;
y == 0;
```

The test script:

```
cvx_begin
variable x;
variable y;
minimize(norm(x + y + 1));
subject to
x == 0;
y == 0;
```

```
cvx_end

fprintf('status:');
disp(cvx_status);

fprintf('optimal value:');
disp(cvx_optval )

fprintf('optimal x:\n');
disp(x)

fprintf('optimal y:\n');
disp(y)
```

The cvxpy test script:

```
import cvxpy as cp

# Create two scalar optimization variables.
x = cp.Variable()
y = cp.Variable()

# Create two constraints.
constraints = [x == 0, y == 0]

# Form objective.
obj = cp.Minimize((x - y + 2)**2)

# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve()  # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", x.value, y.value)
```

(b) square(square(x + y)) <= x - y
this code violates the one of the composition ruleset:
http://web.cvxr.com/cvx/doc/dcp.html#expression-rules
If the function is neither nondecreasing or nonincreasing in an argument, that argument must be affine.

So, use instead power(x + y, 4) <= x - y

The cvx test script:

```
cvx_begin
```

```
variable x;
variable y;
minimize(norm(x + y + 1));
subject to
power(x + y, 4) <= x - y
cvx_end

fprintf('status:');
disp(cvx_status);

fprintf('optimal value:');
disp(cvx_optval )

fprintf('optimal x:\n');
disp(x)

fprintf('optimal y:\n');
disp(y)
```

The cvxpy test script:

```
import cvxpy as cp

# Create two scalar optimization variables.
x = cp.Variable()
y = cp.Variable()

# Create two constraints.
constraints = [(x + y) ** 4 <= x - y]

# Form objective.
obj = cp.Minimize((x - y + 2)**2)

# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve()  # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", x.value, y.value)
```

(c) 1/x + 1/y <= 1;

this expression violates the rule: **a valid call to a function in the atom library with a convex result**
http://web.cvxr.com/cvx/doc/dcp.html#expression-rules

Workaround: use
(c) inv_pos(x) or pow_p(x,-1) instead $1/x$

The cvx test script:

```
cvx_begin
variable x;
variable y;
minimize(norm(x + y + 1));
subject to
inv_pos(x) + inv_pos(y) <= 1;
x >= 0;
y >= 0;
cvx_end

fprintf('status:');
disp(cvx_status);

fprintf('optimal value:');
disp(cvx_optval )

fprintf('optimal x:\n');
disp(x)

fprintf('optimal y:\n');
disp(y)
```

The cvxpy test script:

```
import cvxpy as cp

# Create two scalar optimization variables.
x = cp.Variable()
y = cp.Variable()

# Create two constraints.
constraints = [cp.inv_pos(x) + cp.inv_pos(y) <= 1]

# Form objective.
obj = cp.Minimize((x - y + 2)**2)

# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve()  # Returns the optimal value.
print("status:", prob.status)
```

```
print("optimal value", prob.value)
print("optimal var", x.value, y.value)
```

(d) norm([max(x,1), max(y,2)]) <= 3*x + y

The problem is that norm() is neither nondecreasing or nonincreasing and it can accept only affine arguments.

If the function is neither nondecreasing or nonincreasing in an argument, that argument must be affine
`http://web.cvxr.com/cvx/doc/dcp.html#constraints`)

Workaround: use
z1 >= max(x,1); z2 >= max(y,2) norm([z1, z2]) <= 3*x + y;

The cvx test script:

```
cvx_begin
variable x;
variable y;
minimize(norm(x + y + 1));
subject to
z1 >= max(x,1);
z2 >=  max(y,2)
norm([z1, z2]) <= 3*x + y;
cvx_end

fprintf('status:');
disp(cvx_status);

fprintf('optimal value:');
disp(cvx_optval )

fprintf('optimal x:\n');
disp(x)

fprintf('optimal y:\n');
disp(y)
```

The cvxpy test script:

```
import cvxpy as cp

# Create two scalar optimization variables.
x = cp.Variable()
y = cp.Variable()
z = cp.Variable(2)
z1 = cp.Variable()
```

```
z2 = cp.Variable()

# Create two constraints.
constraints = [z[0] >= cp.maximum(x, 1), z[1] >= cp.maximum(y, 2),
cp.norm(z) <= 3 * x + y]

# Form objective.
obj = cp.Minimize((x - y + 2)**2)

# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve()  # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", x.value, y.value)
```

(e) x*y >= 1; x >= 0; y >= 0

The x*y function is neither convex nor concave, because its Hessian is not positive semidefinite because it has $\lambda_{1,2} = \pm 1$. But we can express this inequality as $x \geq 1/y$ or x >= inv_pos(y). The functions x and $1/y$ are both convex.

The cvx test script:

```
cvx_begin
variable x;
variable y;
minimize(norm(x + y + 1));
subject to
x >= inv_pos(y);
x >= 0;
y >= 0;
cvx_end

fprintf('status:');
disp(cvx_status);

fprintf('optimal value:');
disp(cvx_optval )

fprintf('optimal x:\n');
disp(x)

fprintf('optimal y:\n');
disp(y)
```

The cvx test script:

```
import cvxpy as cp

# Create two scalar optimization variables.
x = cp.Variable()
y = cp.Variable()
z = cp.Variable(2)

# Create two constraints.
constraints = [x >= cp.inv_pos(y), x >= 0, y >= 0]

# Form objective.
obj = cp.Minimize((x - y + 2)**2)

# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve()  # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", x.value, y.value)
```

(f) (x + y)^2/sqrt(y) <= x - y + 5

This is not a **valid call to a function in the atom library with a convex result.** To maid it a valid call we should use the quad_over_lin((x + y), sqrt(y)) function from the atom libary. I do not know why its working in this case.

The cvx test script:

```
cvx_begin
variable x;
variable y;
minimize(norm(x + y + 1));
subject to
quad_over_lin((x + y), sqrt(y)) <= x - y + 5;
y >= 0;
x >= y - 5;
cvx_end

fprintf('status:');
disp(cvx_status);

fprintf('optimal value:');
disp(cvx_optval )

fprintf('optimal x:\n');
```

```
disp(x)

fprintf('optimal y:\n');
disp(y)
```

The cvxpy test script:

```
import cvxpy as cp

# Create two scalar optimization variables.
x = cp.Variable()
y = cp.Variable()

# Create two constraints.
constraints = [cp.quad_over_lin(x + y, cp.sqrt(y)) <= x - y + 5,
y >= 0, x >= y - 5]

# Form objective.
obj = cp.Minimize((x - y + 2)**2)

# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve()  # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", x.value, y.value)
```

(g) x^3 + y^3 ¡= 1; x >= 0; y >= 0

CVX rejects this code because the function x^3 is not convex when x <0. The functions POW_P, POW_POS, or POW_ABS can be used instead.

The cvx test script:

```
cvx_begin
variable x;
variable y;
minimize(norm(x + y + 1));
subject to
pow_p(x, 3) + pow_p(y, 3) <= 1;
x >= 0;
y >= 0;
cvx_end

fprintf('status:');
disp(cvx_status);
```

```
fprintf('optimal value:');
disp(cvx_optval )

fprintf('optimal x:\n');
disp(x)

fprintf('optimal y:\n');
disp(y)
```

The cvx test script:

```
import cvxpy as cp

# Create two scalar optimization variables.
x = cp.Variable()
y = cp.Variable()

# Create two constraints.
constraints = [cp.pos(x) ** 3 + cp.pos(y) ** 3 <= 1, x >= 0, y >= 0]

# Form objective.
obj = cp.Minimize((x - y + 2)**2)

# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve()  # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", x.value, y.value)
```

(h) x + z ¡= 1 + sqrt(x*y - zˆ2); x >= 0; y >= 0

The problem is that x*y is not a concave function. But we can reformulate sqrt(x*y - zˆ2) as sqrt(x * (y - zˆ2 / x)). Then sqrt can be replaced with geo_mean(x, (y - quad_over_lin(z, x)). This works because geo_mean is concave and not decreasing and can accept a concave functions as arguments, and y - quad_over_lin(z, x) is concave.

The test script:

```
cvx_begin
variable x;
variable y;
variable z;
minimize(norm(x + y + z + 1));
```

```
subject to
x + z <= 1 + geo_mean([x, (y - quad_over_lin(z, x))])
x >= 0;
y >= 0;
cvx_end

fprintf('status:');
disp(cvx_status);

fprintf('optimal value:');
disp(cvx_optval )

fprintf('optimal x:\n');
disp(x)

fprintf('optimal y:\n');
disp(y)

fprintf('optimal z:\n');
disp(z)
```

A3.10

3.10 Linear programming with random cost vector. We consider the linear program

$$\begin{array}{l} \text{minimize } c^T x \\ \text{subject to } Ax \preceq 0 \end{array}$$

Here, however, the cost vector c is random, normally distributed with mean $E(c) = c_0$ and covariance $E((c - c_o)(c - c_0)^T) = \Sigma$, ($A$, b, x are deterministic). Thus, for a given $x \in R^n$ the cost $c^T x$ is a scalar random Gaussian variable. We can attach several different meanings to the goal 'minimize $c^T x$; we explore some of these below.

(a) How would you minimize the expected cost $E(c^T x)$ subject to $Ax \preceq 0$?

(b) In general there is a tradeoff between small expected cost and small cost variance. One way to take variance into account is to minimize a linear combination:

$$E(c^T x) + \gamma var(c^T x)$$

of the expected value $E(c^T x)$ and the variance $var(c^T x) = E((c^T x)^2) - E(c^T x)^2$. This is called the 'risk - sensitive cost' and the parameter $\gamma \geq 0$ is called the risk

- aversion parameter since it set the relative values of cost variance and expected value. (For $\gamma > 0$ we are willing to tradeoff an increase in expected cost for a decrease in cost variance). How would you minimize the risk - sensitive cost? Is this problem a convex optimization problem? Be as specifc as you can.

(c) We can also minimize the risk-sensitive cost, but with $\gamma < 0$. This is called 'risk-seeking'. Is this problem a convex optimization problem?

(d) Another way to deal with the randomness in the cost $c^T x$ is to formulate the problem as

$$\begin{array}{l} \text{minimize } \beta \\ \text{subject to } \boldsymbol{prob}(c^T x \geq \beta) \leq \alpha \\ \qquad Ax \preceq 0 \end{array}$$

Here, α is a fixed parameter, which corresponds roughly to the reliability we require, and might typically have a value of 0.01. Is this problem a convex optimization problem? Be as specific as you can. Can you obtain risk-seeking by choice of α? Explain.

Solution:

(a) As x is not a random variable, $E(c^T x) = E(c^T)x = c_0^T x$ and the LP will transform to:

$$\begin{array}{ll} \text{minimize} & c_0^T x \\ \text{subject to} & Ax \preceq 0 \end{array}$$

(b) $E(((c - c_0)x)^2) = E(x^T(c - c_0)(c - c_0)x) = x^T E((c - c_0)(c - c_0)^T)x = x^T \Sigma x$, and the problem will transform to:

$$\begin{array}{ll} \text{minimize} & c_0^T x + \gamma\, x^T \Sigma x \\ \text{subject to} & Ax \preceq 0 \end{array}$$

(c) No, the problem is not convex because the objective function is concave.

(d) Yes, this problem is convex optimization one, see cvxbook p. 158. Lets rewrite the constrain as:

$$\boldsymbol{prob}((\frac{u - \overline{u}}{\Sigma}) \geq \frac{\beta - \overline{u}}{\Sigma}) \leq \alpha$$

Where $u = c^T x$, $\overline{u} = c_0^T x$, $\Sigma^2 = var(c)$ Then the value of $\frac{u-\overline{u}}{\Sigma}$ is a normal variable with unit dispersion and zero mean. Its probability is $\Phi(\frac{\beta-\overline{u}}{\Sigma})$ where

$$\Phi(z) = \int_z^\infty e^{-z^2} dz$$

So, the probability constraint can be expressed as:

$$\Phi(\frac{\beta - \overline{u}}{\Sigma}) \leq \alpha$$

or

$$\frac{\beta - \overline{u}}{\Sigma} \leq \Phi^{-1}(\alpha)$$

or

$$\overline{u} + \Sigma\Phi^{-1}(\alpha) \leq \beta$$

This problem is equivalent to:

$$\begin{array}{ll} \text{minimize} & \beta \\ \text{subject to} & c_0^T x + Q(\alpha)\,||\Sigma^{1/2}x||_2 \leq \beta \\ & Ax \preceq b \end{array}$$

Where $Q(\alpha) = \Phi^-1(\alpha)$ is the quantile of the standard Normal distribution with zero mean and dispersion equal to 1. The problem is convex for $0 \leq \alpha \leq 1/2$, as $Q(\alpha) \leq 0$ for $1/2 \leq \alpha \leq 1$.

We can obtain risk-seeking by choice of α, solving the problem:

$$\begin{array}{c} \text{minimize } -\beta \\ \text{subject to } \boldsymbol{prob}(c^T x \geq \beta) \geq \alpha \\ Ax \preceq 0 \end{array}$$

This problem is equivalent to:

$$\begin{array}{ll} \text{minimize} & -\beta \\ \text{subject to} & c_0^T x - Q(\alpha)\,||\Sigma^{1/2}x||_2 \leq \beta \\ & Ax \preceq b \end{array}$$

and it is convex for $1/2 \leq \alpha \leq 1$.

A3.32

Satisfying a minimum number of constraints. Consider the problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0 \text{ holds for at least } k \text{ values of } i \end{array}$$

with variable $x \in R^n$, where thye objective f_0 and the constraint functions f_i, $i = \{1...m\}$ (with $m \geq k$ is convex. Here we require than only k constraints holds, instead all m of them. In general this is hard combinatorial problem; the brute force solution is to solve all $\binom{m}{k}$ convex problems obtained by choosing subsets of k constraints to impose and selecting one with possible smallest objective value.

In this problem we explore a convex restriction that can be an effective heuristic for the problem.

(a) Suppose $\lambda > 0$. Show than constraint

$$\sum_{i=1}^{m}(1 + \lambda f_i(x))_+ \leq m - k$$

guarantees that $f_i(x) \leq 0$ holds for at least k values of i. ($(u)_+$ means $max\{u, 0\}$.) Hint: for each $u \in R$, $(1 + \lambda u)_+ \geq 1(u\ geq 0)$, where $1(u > 0) = 1$ for $u > 0$ and $1(u > 0) = 0$ for $u \leq 0$.

(b) Consider the problem:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & \sum_{i=1}^{m}(1 + \lambda f_i(x))_+ \leq m - k \\ & \lambda > 0 \end{array}$$

with variables x and λ. This is a restriction of the original problem: if (x, λ) are feasible for it, then x is feasible for iriginal problem. Show how to solve this problem usiong convex optimization. (This may involve a change of variables)

(c) Apply a method in part (b) to the problem instance:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & a_i^T x \leq b_i \text{ holds for at least } k \text{ values of } i \end{array}$$

with $m = 70$, $k = 58$ and $n = 12$. The vectors b, c and the matrix A are given in the file satisfy_some_constraints_data.*. Report the optimal value of λ, the objective value, and an actual value of constraints that are satisfied (which should be large than or equal to k). To determine if a constraint is satisfied, you can use the tolerance $a_i^T x - b_i \leq e^{feas}$, with $e^{feas} = 10^{-5}$.
A standard trick is to take the tentative solution, choose the k constraints with the smallest values of $f_i(x)$ and then minimize $f_0(x)$ subject to these k constraints (i.e ignorint the other $m - k$ constraints). This improves the objective value over the one found using the restriction. Carry this out for the problem instance, and report the objective value obtained.

Solution:

(a)

As

$$\sum_{i=1}^{m}(1 + \lambda f_i(x))_+ \geq \sum_{i=1}^{m} 1(f_i(x) > 0)$$

then

$$\sum_{i=1}^{m} 1(f_i(x) > 0) \leq m - k$$

Considering the definition of $1(u)$, the upper equation means that the number of negative values of $f_i(x)$ in the sum is equal at least to k (or more).

(b)

Changing the variables λ for $1/y$ and multiplying both part of the constraint on y (remembering that $\lambda > 0$, therefore $y > 0$) we get a modified problem:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & \sum_{i=1}^{m}(y + f_i(x))_+ \leq (m-k)y \\ & y > 0 \end{array}$$

as the function $y + f_i(x)$ is convex and nondecreasing and the function $(u)+$ is convex then the $(y + f_i(x))+$ is convex.

The optimal values for : $\lambda = 282.9842$, objective $= -8.4545$, actual value of constraints that are satisfied is 66. Final objective value is -8.75677.

the matlab code:

```
satisfy_some_constraints_data;

cvx_begin
```

```
variable x(n);
variable y;
minimize (c.' * x);
subject to
sum(pos(y +  A * x - b)) <= (m - k) * y;
y > 0;
cvx_end

fprintf('status = ');
disp(cvx_status);

fprintf('optimal value = ');
disp(cvx_optval )

fprintf('optimal x = \n');
disp(x)

fprintf('optimal lambda = ');
disp(1 / y)

e_feas = 1e-5;
fprintf('actual constraints safisfied = ')
disp(sum(A * x - b <= e_feas))

% get inedxes of k firt minimal
[~, idx] = (sort(A * x - b));
min_idx = idx(1:k);
% get the rows of A corresponding these minimal indexes
A_min = A(min_idx, :);
% get the elements of b corresponding these minimal indexes
b_min = b(min_idx);

cvx_begin
variable x(n);
minimize (c.' * x);
subject to
A_min * x <= b_min;
cvx_end

fprintf('final objective value = %f', cvx_optval)
```