EE 364a HW7

Yen-Yu Chang
(discuss with Cheng-Min Chiang)

February 28, 2020

8.11 Since all cones contain 0, we can assume that each a_i is nonzero in our discussion. We know that the points lie in some Euclidean cone if and only if the point lie in some halfspace. And the largest Euclidean cone has angular radius as $\pi/2$, then we have the linear inequalities as:

$$a_i^T x \geq 0, \quad i = 1, \dots m.$$

Now we have to find the smallest Euclidean cone. We know that the points lie in a cone of angular radius θ if and only if there is a vector $x \in \mathbf{R}^n$ such that

$$\frac{a_i^T x}{||a_i||_2 ||x||_2} \geq \cos\theta, \quad i = 1, \dots, m.$$

which is a set of second-order cone constraints. Therefore, we can use binary search on θ to find the smallest cone as solving a sequence of SOCP feasibility problems.

8.15 Let $A^{-1} = \tilde{A}^T\tilde{A}$, then we parametrize the ellipsoid $\mathcal{E}$ as

$$\mathcal{E} = \left\{ x | x^T \tilde{A}^T \tilde{A} x + 2\left(\tilde{A}^T(-\tilde{A}x_0)\right)^T x + (-\tilde{A}x_0)^T(-\tilde{A}x_0) - 1 \leq 0 \right\}.$$

Now we use the result from Appendix B.2, that $\mathcal{E}_i \subseteq \mathcal{E}$ if and only if there exists a $\tau \geq 0$ such that

$$\begin{bmatrix} \tilde{A}^2 - \tau A_i & \tilde{A}(-\tilde{A}x_0) - \tau b_i \\ \left(\tilde{A}(-\tilde{A}x_0) - \tau b_i\right)^T & (-\tilde{A}x_0)^T(-\tilde{A}x_0) - 1 - \tau c_i \end{bmatrix} \preceq 0$$

The volume of $\mathcal{E}$ is proportional to $\det \tilde{A}^{-1}$, so we can find the minimum volume ellipsoid that contains $\mathcal{E}_1, \dots, \mathcal{E}_m$ by solving

$$\begin{array}{ll} \text{minimize} & \log\det \tilde{A}^{-1} \\ \text{subject to} & \begin{bmatrix} \tilde{A}^2 - \tau_i A_i & \tilde{A}(-\tilde{A}x_0) - \tau_i b_i \\ \left(\tilde{A}(-\tilde{A}x_0) - \tau_i b_i\right)^T & (-\tilde{A}x_0)^T(-\tilde{A}x_0) - 1 - \tau_i c_i \end{bmatrix} \preceq 0, \quad i = 1, \dots, K \\ & \tau_i \geq 0, \quad i = 1, \dots, K. \end{array}$$

which is also equal to

$$\begin{array}{ll}\text{minimize} & \log\det \tilde{A}^{-1} \\ \text{subject to} & \begin{bmatrix} \tilde{A}^2 - \tau_i A_i & \tilde{A}(-\tilde{A}x_0) - \tau_i b_i & 0 \\ \left(\tilde{A}(-\tilde{A}x_0) - \tau_i b_i\right)^T & -1 - \tau_i c_i & (-\tilde{A}x_0)^T \\ 0 & (-\tilde{A}x_0) & -\tilde{A}^2 \end{bmatrix} \preceq 0, \quad i = 1, \dots, K \\ & \tau_i \geq 0, \quad i = 1, \dots, K. \end{array}$$

9.1

(a) If $P \not\succeq 0$, which means that there exists a v such that $v^T P v < 0$. Then we can set $x = tv$ and we get

$$f(x) = \frac{t^2 v^T P v}{2} + t q^T v + r,$$

which goes to $-\infty$ while t is large.

(b) It implies that q is not in the range of P. Then we can express q as $q = \tilde{q} + v$, where $\tilde{q}$ is the Euclidean projection onto $\mathcal{R}(P)$ and take $v = q - \tilde{q}$. This means that v is nonzero and orthogonal to $\mathcal{R}(P)$ *i.e.*, $v^T P v = 0$. If we let $x = tv$, then we have the following equation:

$$f(x) = (1/2)(tv)^T P tv + q^T(tv) + r = tq^v + r = t(\tilde{q} + v)^T v + r = t(v^T v) + r,$$

which goes to $-\infty$ while t is large.

A6.31

(a) If $z_y \leq z_j$ for some index $j \neq y$, we can express $\sum_{j=1}^k \exp(z_j - z_y)$ as

$$\sum_{j=1}^k \exp(z_j - z_y) = \sum_{\{j \mid z_y \leq z_j, j \neq y\}} \exp(z_j - z_y) + \sum_{\{j \mid z_y > z_j, j \neq y\}} \exp(z_j - z_y) + \sum_{\{j \mid j = y\}} \exp(z_j - z_y)$$

For those $j \neq y$ and $z_y \leq z_j$, we have $\exp(z_j - z_y) \geq 1$ which means that

$$\sum_{\{j \mid z_y \leq z_j, j \neq y\}} \exp(z_j - z_y) \geq 1.$$

And we know that $\sum_{\{j \mid z_y > z_j, j \neq y\}} \exp(z_j - z_y) \geq 0$, $\sum_{\{j \mid j = y\}} \exp(z_j - z_y) = 1$. Therefore, $\sum_{j=1}^k \exp(z_j - z_y) \geq 2$ which means that if $z_y \leq z_j$ for some index $j \neq y$,

$$l_{\mathrm{mc}(z,y)} = \log\left(\sum_{j=1}^k \exp(z_j - z_y)\right) \geq \log 2 = 1.$$

Suppose we do not have any constraint, we have

$$\sum_{j=1}^{k} \exp\left(z_j - z_y\right) \geq 1$$

because there is an index $j = y$. Therefore,

$$l_{\mathrm{mc}}(z, y) = \log\left(\sum_{j=1}^{k} \exp\left(z_j - z_y\right)\right) \geq 0.$$

It means that $\inf_z l_{\mathrm{mc}}(z, y) = 0$ if $z_j = -\infty$ for all $j \neq y$.

(b) Let $z = \Theta x$ and z_i is the i'th column of z, we can reformulate the problem as

$$\begin{array}{ll} \text{minimize} & \sum\limits_{i=1}^{m} l_{\mathrm{mc}}(z_i, y_i) + \lambda||\Theta|| \\ \text{subject to} & z = \Theta x \end{array}$$

Then we can get the Lagrangian as

$$L(\Theta, z, \nu) = \sum_{i=1}^{m} l_{\mathrm{mc}}(z_i, y_i) + \lambda||\Theta|| + \mathbf{tr}\left(\nu^T(\Theta x - z)\right)$$

And the dual function is

$$g(\nu) = \inf_{\Theta, z}\left(\sum_{i=1}^{m} l_{\mathrm{mc}}(z_i, y_i) + \lambda||\Theta|| + \mathbf{tr}(\nu^T\Theta x) - \mathbf{tr}(\nu^T z)\right)$$

we can first compute the infimum on Θ, then we get

$$g(\nu) = \begin{cases} \inf\limits_{z}\left(\sum\limits_{i=1}^{m} l_{\mathrm{mc}}(z_i, y_i) - \mathbf{tr}(\nu^T z)\right) & ||x^T\nu||_* \leq \lambda \\ -\infty & \text{otherwise} \end{cases}$$

Then we compute the infimum on z, we get

$$g(\nu) = \begin{cases} -\sum\limits_{i=1}^{m}\left(\sum\limits_{j=1, j\neq y_i}^{k} \nu_{ji}\log\nu_{ji} - (1+\nu_{y_i i})\log(1+\nu_{y_i i})\right) & ||x^T\nu||_* \leq \lambda, \mathbf{1}^T\nu_i = 0 \\ -\infty & \text{otherwise} \end{cases}$$

Therefore, the dual problem is

$$\begin{array}{ll} \text{maximize} & -\sum_{i=1}^{m}\left(\sum_{j=1,j\neq y_i}^{k}\nu_{ji}\log\nu_{ji}+(1+\nu_{y_i i})\log(1+\nu_{y_i i})\right) \\ \text{subject to} & ||x^T\nu||_* \leq \lambda \\ & \mathbf{1}^T\nu_i = 0, \quad i=1,\ldots,m \end{array}$$

(c) Let $z = \Theta x$ and z_i is the i'th column of z, then we can reformulate the problem as

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m} l_{\text{mc}}(z_i, y_i) + \lambda\sum_{j=1}^{k}||\theta_j|| \\ \text{subject to} & z = \Theta x \end{array}$$

And the Lagrangian is

$$L(\Theta, z, \nu) = \sum_{i=1}^{m} l_{\text{mc}}(z_i, y_i) + \lambda\sum_{j=1}^{k}||\theta_j|| + \mathbf{tr}(\nu^T\Theta x) - \mathbf{tr}(\nu^T z)$$

Then we can reformulate the Lagrangian as

$$L(\theta_1,\ldots,\theta_k,z,\nu) = \sum_{i=1}^{m} l_{\text{mc}}(z_i, y_i) - \mathbf{tr}(\nu^T z) + \sum_{j=1}^{k}\left(\lambda||\theta_j|| + (\sum_{i=1}^{m}\nu_{ji}x_i^T)\theta_j\right)$$

The dual function is

$$g(\nu) = \inf_{\theta_1,\ldots,\theta_k,z}\left(\sum_{i=1}^{m} l_{\text{mc}}(z_i, y_i) - \mathbf{tr}(\nu^T z) + \sum_{j=1}^{k}(\lambda||\theta_j|| + (\sum_{i=1}^{m}\nu_{ji}x_i^T)\theta_j\right)$$

We first compute the infimum on each θ_i, then we have

$$g(\nu) = \begin{cases} \inf_z\left(\sum_{i=1}^{m} l_{\text{mc}}(z_i, y_i) - \mathbf{tr}(\nu^T z)\right) & ||(\sum_{i=1}^{m}\nu_{ji}x_i^T)^T||_* \leq \lambda, \quad j=1,\ldots,k \\ -\infty & \text{otherwise} \end{cases}$$

Then we compute the infimum on ν, we get

$$g(\nu) = \begin{cases} -\sum_{i=1}^{m}\left(\sum_{j=1,j\neq y_i}^{k}\nu_{ji}\log\nu_{ji}+(1+\nu_{y_i i})\log(1+\nu_{y_i i})\right) & ||(\sum_{i=1}^{m}\nu_{ji}x_i^T)^T||_* \leq \lambda, j=1,\ldots,k \\ & \mathbf{1}^T\nu_i = 0, i=1,\ldots,m \\ -\infty & \text{otherwise} \end{cases}$$

Therefore, the dual problem is

$$\begin{array}{ll}\text{maximize} & -\sum\limits_{i=1}^{m}\left(\sum\limits_{j=1,j\neq y_i}^{k}\nu_{ji}\log\nu_{ji}+(1+\nu_{y_ii})\log(1+\nu_{y_ii})\right)\\ \text{subject to} & ||(\sum\limits_{i=1}^{m}\nu_{ji}x_i^T)^T||_*\leq\lambda, j=1,\dots,k\\ & \mathbf{1}^T\nu_i=0, i=1,\dots,m\end{array}$$

(d) Let $z=\Theta x$, then we can formulate the problem as

$$\begin{array}{ll}\text{minimize} & \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{k}\phi(z_{ji}-z_{y_ii})+\lambda||\Theta||\\ \text{subject to} & z=\Theta x\end{array}$$

Then we have the Lagrangian as

$$L(\Theta,z,\nu)=\sum_{i=1}^{m}\sum_{j=1}^{k}\phi(z_{ji}-z_{y_ii})+\lambda||\Theta||+\mathbf{tr}\left(\nu^T(\Theta x-z)\right)$$

It is equal to

$$L(\Theta,z,\nu)=\sum_{i=1}^{m}\sum_{j=1}^{k}\phi(z_{ji}-z_{y_ii})-\mathbf{tr}(\nu^Tz)+\lambda||\Theta||+\mathbf{tr}(\nu^T\Theta x)$$

And the dual function is

$$g(\nu)=\inf_{\Theta,x}\left(\sum_{i=1}^{m}\sum_{j=1}^{k}\phi(z_{ji}-z_{y_ii})-\mathbf{tr}(\nu^Tz)+\lambda||\Theta||+\mathbf{tr}(\nu^T\Theta x)\right)$$

We first compute the infimum on Θ, then we get

$$g(\nu)=\begin{cases}\inf\limits_{z}\left(\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{k}\phi(z_{ji}-z_{y_ii})-\mathbf{tr}(\nu^Tz)\right) & , \quad ||x^T\nu||_*\leq\lambda\\ -\infty & , \quad \text{otherwise}\end{cases}$$

It is equal to

$$g(\nu)=\begin{cases}\inf\limits_{z}\left(\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{k}(\phi(z_{ji}-z_{y_ii})-\nu_{ji}(z_{ji}-z_{y_ii}))\right)-\inf\limits_{z}\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{k}\nu_{ji}z_{y_ii} & , \quad ||x^T\nu||_*\leq\lambda\\ -\infty & , \quad \text{otherwise}\end{cases}$$

And $\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{k}\nu_{ji}z_{y_i i}$ has to be bounded below, so $\sum_{j=1}^{k}\nu_{ji}=0$. Hence, the dual function becomes

$$g(\nu)=\begin{cases} m\phi(0)-\sum\limits_{i=1}^{m}\sum\limits_{j=1,j\neq y_i}^{k}\phi^*(\nu_{ji}) & , \quad \sum\limits_{j=1}^{k}\nu_{ji}=0, ||x^T\nu||_*\leq\lambda \\ -\infty & , \quad otherwise \end{cases}$$

Therefore, the dual problem is

$$\begin{array}{ll} \text{maximize} & m\phi(0)-\sum\limits_{i=1}^{m}\sum\limits_{j=1,j\neq y_i}^{k}\phi^*(\nu_{ji}) \\ \text{subject to} & ||x^T\nu||_*\leq\lambda \\ & \sum\limits_{j=1}^{k}\nu_{ji}=0, i=1,\dots,m \end{array}$$

A8.5

(a) We can reformulate

$$\sum_{i=1}^{k}(a_i^Tx-b_i)^2+\delta\sum_{i=1}^{n-1}(x_i-x_{i+1})^2+\eta\sum_{i=1}^{n}x_i^2$$

as

$$(Ax-b)^T(Ax-b)+\delta(\Delta x)^T(\Delta x)+\eta x^Tx$$

which is equal to

$$x^TA^TAx-2b^TAx+b^Tb+\delta x^T\Delta^T\Delta x+\eta x^Tx$$

where

$$\Delta=\begin{bmatrix} 1 & -1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & -1 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 & -1 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix}$$

And the optimal conditions is:

$$\nabla\left(x^TA^TAx-2b^TAx+b^Tb+\delta x^T\Delta^T\Delta x+\eta x^Tx\right)=0$$

which means that

$$(A^TA+\delta\Delta^T\Delta+\eta I)x^*=A^Tb$$

We can reset $\Delta=\Delta^T\Delta$, then the optimality condition becomes $(A^TA+\delta\Delta+\eta I)x^*=A^Tb$

(b) If no structure is exploited, we can solve the problem in $\frac{1}{3}n^3$ flops approximately. Since $k \ll m$, we can apply Sherman-Morrison-Woodbury formula as

$$x^* = (\delta\Delta + \eta I)^{-1}A^T b - (\delta\Delta + \eta I)^{-1}A^T \left(I + A(\delta\Delta + \eta I)^{-1}A^T\right)^{-1} A(\delta\Delta + \eta I)^{-1}A^T x$$

to efficiently solve the problem as follow:

(i) Solve $(\delta\Delta+\eta I)z_1 = A^T b$ and $(\delta\Delta+\eta I)Z_2 = A^T$ for z_1 and Z_2. Because of $\delta\Delta+\eta I$ is tridiagonal, we can solve $(\delta\Delta+\eta I)z_1 = A^T b$ in $O(n)$ and $(\delta\Delta+\eta I)Z_2 = A^T$ in $O(nk)$ since $A^T b \in \mathbf{R}^n$ and $A^T \in \mathbf{R}^{n\times k}$.

(ii) Form Az_1 and AZ_2 need $2nk$ and $2nk^2$ flops.

(iii) Solve $(I + AZ_2)z_3 = Az_1$ for z_3 needs $\frac{1}{3}k^3$ flops since $Az_1 \in \mathbf{R}^k$.

(iv) Form $x^* = z_1 - Z_2 z_3$ needs $n + 2nk$ flops.

The total flops is in $O(nk^2)$, which is much smaller than $(1/3)n^3$ when $k \ll n$.

(c) The following MATLAB code solves the problem:

```
n = 4000;
k = 100;
delta = 1;
eta = 1;

A = rand(k, n);
b = rand(k, 1);

e = ones(n, 1);
D = spdiags([-e 2*e -e],[-1 0 1], n,n);
D(1, 1) = 1;
D(n, n) = 1;
I = sparse(1:n ,1:n ,1);
F = A' * A + eta * I + delta * D;
P = eta * I + delta * D;
g = A' * b;

fprintf('\nComputing solution directly\n');
s1 = cputime;
x_gen = F \ g;
s2 = cputime;
fprintf('Done (in %g sec)\n',s2-s1);


fprintf('\nComputing solution using efficient method\n');
```

```
%x_eff = P^{-1}g - P^{-1}A'(I +AP^{-1}A')^{-1}AP^{-1}g.
t1= cputime;
Z_0 = P \ [g A'];
z_1 = Z_0(:, 1);
Z_2 = Z_0(:, 2:k+1);
z_3 = (sparse(1:k, 1:k, 1) + A * Z_2) \ (A * z_1);
x_eff = z_1 - Z_2 * z_3;
t2 = cputime;
fprintf('Done (in %g sec)\n',t2-t1);
fprintf('\nrelative error = %e\n',norm(x_eff-x_gen)/norm(x_gen));
```

Then we see that it needs 67.53 seconds to compute the solution directly and it needs 2.89 seconds to compute the solution by our efficient method.

A.11.3 We know that the noise level can be expressed recursively as:

$$N_0^2 = 0, \quad N_i^2 = a_i(N_{i-1}^2 + \alpha_i^2), \quad i = 1, \dots, n$$

Then we can reformulate the square of output noise level as:

$$N_{out}^2 = N_n^2 = \sum_{i=1}^{n} \prod_{j=i}^{n} a_j^2 \alpha_i^2$$

And the amplifier overload constraints can be expressed as:

$$S_i = S_{in} \prod_{j=1}^{i} a_j \le M_i, \quad i = 1, \dots, n.$$

Therefore, the maximum input signal level is:

$$S_{in,max} = \min_i \frac{M_i}{\prod_{j=1}^{i} a_j}.$$

Since $S_n = A^{tot} S_{in}$, we can express the maximum output signal level as:

$$S_{max} = A^{tot} S_{in,max} = \min_i M_i \prod_{j=i+1}^{n} a_j.$$

Hence, the dynamic range can be expressed as:

$$D = \frac{\min\limits_i M_i \prod\limits_{j=i+1}^{n} a_j}{(\sum\limits_{i=1}^{n} \prod\limits_{j=1}^{n} a_j^2 \alpha_i^2)^{1/2}}$$

The original problem to maximize D can be expressed as:

$$
\begin{array}{ll}
\text{maximize} & \dfrac{\min\limits_i M_i \prod\limits_{j=i+1}^{n} a_j}{(\sum\limits_{i=1}^{n}\prod\limits_{j=1}^{n} a_j^2\alpha_i^2)^{1/2}} \\
\text{subject to} & a_1 \dots a_n = A^{tot} \\
 & a_i \le A_i, \quad i = 1, \dots, n.
\end{array}
$$

Instead of maximizing D, we can minimize $1/D^2$ and it is a posynomial function, which makes the problem become a geometric programming.

$$
\begin{array}{ll}
\text{minimize} & \dfrac{(\sum\limits_{i=1}^{n}\prod\limits_{j=1}^{n} a_j^2\alpha_i^2)}{\min\limits_i M_i^2 \prod\limits_{j=i+1}^{n} a_j^2} \\
\text{subject to} & a_1 \dots a_n = A^{tot} \\
 & a_i \le A_i, \quad i = 1, \dots, n.
\end{array}
$$

To make the problem more simple, we can introduce a new variable t, and rewrite the optimization problem as

$$
\begin{array}{ll}
\text{minimize} & t \\
\text{subject to} & a_1 \dots a_n = A^{tot} \\
 & a_i \le A_i, \quad i = 1, \dots, n \\
 & \sum\limits_{i=1}^{n}\prod\limits_{j=i}^{n} a_j^2\alpha_i^2 \le tM_i^2 \prod\limits_{j=i+1}^{n} a_j^2, \quad i = 1, \dots, n.
\end{array}
$$

And the following Python code solves the problem:

```
import numpy as np
import cvxpy as cvx

n = 4
A_tot = 10000
alpha = cvx.Constant([0.00001, 0.01, 0.01, 0.01])
M = cvx.Constant([0.1, 5, 10, 10])
A_max = cvx.Constant([40, 40, 40, 20])

a = cvx.Variable(n, pos = True)
t = cvx.Variable(1, pos = True)

obj = cvx.Minimize(t)

constraints = [cvx.prod(a) == A_tot,\
  a <= A_max]
```

```
left = cvx.Constant(1e-10)

for i in range(n):
tmp = alpha[i] ** 2
for j in range(i, n):
tmp = cvx.multiply(tmp,cvx.square(a[j]))
left = left + tmp

for i in range(n - 1):
right = 1.0
for j in range(i + 1, n):
right = cvx.multiply(right, cvx.square(a[j]))
right = cvx.multiply(right, cvx.square(M[i]))
constraints += [left <= t * right]

constraints += [left <= t * M[3] * M[3]]

problem = cvx.Problem(cvx.Minimize(t), constraints)
problem.solve(gp=True, solver = cvx.ECOS)

print("The optimal gains are: {}".format(a.value))
print("The optimal dynamic range is: {}".format(t.value ** -0.5))
```

Then we get the optimal gains $39.99, 39.99, 6.248, 1.000$ and the optimal dynamic range 3.9955.