

HW7

Carl Mueller
CSCI 5254 - Convex Optimization

April 25, 2018

8.16

Formulate the following as a CVX optimization problem:
Find the rectangle

$$R = \{ x \in \mathbf{R}^n \mid l \preceq x \preceq u \}$$

of maximum volume enclosed in the polyhedron

$$P = \{ x \mid Ax \preceq b \}$$

The volume can be expressed as:

Proposition 1.

$$v = \prod_{i=1}^n u_i - l_i \tag{1}$$

We want all the 2^n corners to be contained within the polyhedron. This every corner must meet the polyhedron constraint $Ac \preceq b$. Where c is the vector of corners. Each of these corners can be more succinctly represented as the vector based on the upper and lower values of each edge:

If we express x_i as $u_i - l_i$ then this system becomes

$$\sum_{i=1}^n a_{ij}(u_j - l_j) \leq b_i$$

The problem can be expressed as:

$$\begin{aligned} & \text{minimize} && \prod_{i=1}^n u_i - l_i \\ & \text{subject to} && \sum_{i=1}^n a_{ij}(u_j - l_j) \leq b_i \end{aligned}$$

The constraint is a posynomial as it is a summation of the monomial $a_{ij}(u_j - l_j)$.
To make the problem a non-linear geometric optimization problem, we take the log of the objective:

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n \log(u_i - l_i) \\ & \text{subject to} && \sum_{i=1}^n a_{ij}(u_j - l_j) \leq b_i \end{aligned}$$

8.24

We make use of the Cauchy-Schwarz inequality and substitute p knowing that $\|u\|_2 \leq p$:

Proposition 1.

$$u^T x_i \leq \|u\|_2 \|x_i\|_2 \quad (2)$$

$$\|u\|_2 \|x_i\|_2 \leq p \|x_i\|_2 \quad (3)$$

$$u^T y_j \leq \|u\|_2 \|y_j\|_2 \quad (4)$$

$$\|u\|_2 \|y_j\|_2 \leq p \|y_j\|_2 \quad (5)$$

$$-\|u\|_2 \|y_j\|_2 \geq -p \|y_j\|_2 \quad (6)$$

$$(7)$$

For x_i :

$$(a + u)^T x_i \geq b$$

$$a^T x_i + u^T x_i \geq b$$

$$a^T x_i + \|u\|_2 \|x_i\|_2 \geq b$$

$$a^T x_i + p \|x_i\|_2 \geq b$$

$$a^T x_i - b \geq -p \|x_i\|_2$$

For y_j :

$$(a + u)^T y_j \leq b$$

$$a^T y_j + u^T y_j \leq b$$

$$a^T y_j + \|u\|_2 \|y_j\|_2 \leq b$$

$$a^T y_j + p \|y_j\|_2 \leq b$$

$$a^T y_j - b \leq -p \|y_j\|_2$$

$$b - a^T y_j \geq p \|y_j\|_2$$

The optimization problem:

$$\begin{aligned} & \text{minimize} && p \\ & \text{subject to} && b - a^T y_j \geq p \|y_j\|_2 \\ & && a^T x_i - b \geq -p \|x_i\|_2 \\ & && \|a\|_2 \leq 1 \end{aligned}$$

Additional Exercises:

5.12

One heuristic estimate an initial \hat{x} using the huber penalty function. We then use that \hat{x} to estimate a \hat{P} by aligning the indices of Ax and y to find a permutation matrix. Then using that same permutation we reoptimized for \hat{x} . We repeat this algorithm until the euclidean norm of the distance between the \hat{x}_τ and $\hat{x}_{\tau-1}$ is below some tolerance, τ being the current iteration step. **Code:**

```
above_tol = 1
tolerance = .00000001
% Seed our initial estimate of x using huber function
cvx_begin
variable x(n);
    minimize( sum(huber(A*x-y)) );
cvx_end
P_hat = eye(m)
x_prior = zeros(n)
while 1

    % Align the smallest indexes, find pi (the permutation index alignment)
    % and construct the permutation matrix P_hat accordingly:
    [Ax_values, Ax_idx] = sort(A*x);
    [y_values, y_idx] = sort(y);
    pi = [y_idx' ; Ax_idx'];
    P_temp = zeros(m, m);
    for i = 1 : m
        row = pi(1,i);
        col = pi(2,i);
        P_temp(row, col) = 1;
    end
    P_hat = P_temp;
    if P_hat*P_hat' ~= eye(m)
        "Invalid P_hat!"
        break
    end
    "Distance:"
    dist = norm(x - x_prior, 2)
    if dist <= tolerance
        break
    end
end
```

```

x_prior = x;

% Find x_hat
cvx_begin
    variable x(n,1)
    minimize(norm(A*x-P_hat'*y, 2))
cvx_end;
end
P_eye = eye(m);
cvx_begin
    variable x_eye(n,1)
    minimize(norm(A*x_eye-P_eye'*y, 2))
cvx_end;
"Distance x (P=I) and estimated x:"
norm(x_eye - x_true, 2)
"Distance x_true and estimated x:"
norm(x_true - x, 2)

Results:
"Distance estimated x (P=I) and x_true:"
ans = 3.4363
"Distance x_true and estimated x:"
ans = 0.0965

```

5.18

We can reformulate the problem as the original object being less than or equal to some value z :

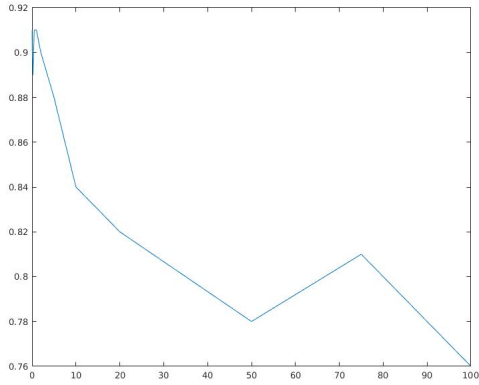
$$1 + \max_{k \neq y_i} f_k(x_i) - f_{y_i}(x_i) \leq z_i, z_i \geq 0$$

This can be represented by the following problem:

$$\begin{aligned}
 & \text{minimize} && \sum_i z_i + \mu \|A\|_F^2 \\
 & \text{subject to} && 1 + \max_{k \neq y_i} f_k(x_i) - f_{y_i}(x_i) \leq z_i, \forall i \\
 & && 1^T b = 0, z \geq 0
 \end{aligned}$$

This can be reexpressed using the individual inequality constraints:

$$\begin{aligned}
 & \text{minimize} && \sum_i z_i + \mu \|A\|_F^2 \\
 & \text{subject to} && 1^T b = 0, z \geq 0 \\
 & && 1 + a_k^T x_i + b - y_i \leq z_i, k = 1, 2, \dots, y_i - 1, y_i + 1, \dots, K, i = 1, 2, \dots, m
 \end{aligned}$$



Code:

```

E = [];
U = [0.01 0.05 0.1 0.2 0.5 1 2 5 10 20 50 75 100]
% This loop generates a new u value.
for u = 1:size(U,2)
    cvx_begin
        variable z(mTrain, 1)
        variable A(K, n)
        variable b(K, 1)
        minimize(sum(z) + U(u)*square_pos(norm(A, 'fro')))
        subject to
            for i=1:mTrain
                for k=[1:y(i)-1 y(i)+1:K]
                    1+(A(k,:)*x(:,i)+b(k))-(A(y(i),:)*x(:,i)+b(y(i))) <= z(i);
                end
                z(i) >= 0;
            end
            sum(b) == 0;
    cvx_end
    % Compute the predicted labels by computing the affine function
    % on xtest using the estimated optimal A and b. Find the max in each
    % column (i.e. argmax for label) and round to get whole number value.
    correct = 0
    y_pred = zeros(1,mTest);

    for i=1:mTest
        [~, y_pred(i)] = max(A*xtest(:,i) + b);
        if (y_pred(i) == ytest(i))
            correct = correct + 1;
        end
    end
end

```

```

        end
    end
    percent_correct = correct/mTest
    E = [E ; percent_correct]
end
plot(U,E)

```

13.15

One heuristic is to ensure that the 1-norm of w is minimized. We can then formulate an optimization problem subject to the following constraint:

Proposition 1.

$$E[(r - \bar{r})(r - \bar{r})] = \Sigma \quad (8)$$

$$E[rr^T] - \bar{r}\bar{r}^T = \Sigma \quad (9)$$

$$E[rr^T] = \Sigma + \bar{r}\bar{r}^T \quad (10)$$

Proposition 2.

$$E[z^T z] = c^T \text{diag}(\Sigma) + \bar{r}^T c c^T \bar{r} \quad (11)$$

$$E[(z - w^T r)(z - w^T r)] \leq .01 E[z^2]$$

$$E[z^T z + r^T w w^T r - 2z w^T r] \leq .01 E[z^2]$$

$$E[z^T z] + E[r^T w w^T r] - 2E[z w^T r] \leq .01 E[z^2]$$

$$E[z^T z] + E[r^T w w^T r] - 2E[z w^T r] \leq .01 E[z^2]$$

$$E[z^T z] + E[w^T r r^T w] - 2E[(c^T r)^T w^T r] \leq .01 E[z^2]$$

$$c^T \text{diag}(\Sigma) + \bar{r}^T c c^T \bar{r} + w^T (\Sigma + \bar{r}\bar{r}^T) w - 2E[(c^T r)^T w^T r] \leq .01 E[z^2]$$

$$c^T \text{diag}(\Sigma) + \bar{r}^T c c^T \bar{r} + w^T (\Sigma + \bar{r}\bar{r}^T) w - 2w^T (\Sigma + \bar{r}\bar{r}^T) c \leq .01 E[z^2]$$

This becomes the optimization problem reflected in the matlab code:

```

minimize    ||w||1
subject to  c^T diag(Sigma) + rbar^T c c^T rbar + w^T (Sigma + rbar rbar^T) w - 2w^T (Sigma + rbar rbar^T) c <= .01 E[z^2]

```

Code:

```

ctc = mtimes(c', c);
rbarsq = dot(rbar, rbar');
zsqr = (ctc * rbarsq)
cvx_begin
    variable w(n)
    minimize norm(w,1)

```

```

E_num = ( (rbar' * (c * c') * rbar)) + c'*diag(Sigma) + (w' * (Sigma + (rba
subject to
    E_num <= 0.01 * (ctc * rbarsq);
    w <= c;
cvx_end
E_num/(ctc * rbarsq)
sum(abs(w > 0.01))
sum(abs(c > 0.01))
Results:
ans = 0.0100
ans = 108
ans = 500

```

16.5

We begin by stating that $X = \theta s$. The optimization problem can be formulated as:

$$\begin{aligned}
 & \text{minimize} && \sum_{t=1}^T \phi(s_t) \\
 & \text{subject to} && S_{min} \preceq s \preceq S_{max} \\
 & && |s_{t+1} - s_t| \leq R \\
 & && s = X1, X^T 1 \succeq W, X \succeq 0 \\
 & && X_{ti} = 0, t = 1, \dots, A_i - 1, 1 = i, \dots, n \\
 & && X_{ti} = 0, t = D_i + 1, \dots, T, 1 = i, \dots, n
 \end{aligned}$$

The top two constraints are the processor speed limits and slew rates. Each row coefficients θ_{ti} for $1 = 1, \dots, n$ must sum to one therefore $X1 = s$. The last two constraints express that each component of X_{ti} is 0 for any values outside the range $[A_i, D_i]$ for the given job i . **Code:**

```

cvx_begin
    variable X(T,n)
    s = sum(X');
    minimize(sum(alpha+beta*s+gamma*square(s)))
    subject to
        X >= 0;
        Smin <= s <= Smax;
        abs(s(2:end)-s(1:end-1))<=R; % slew rate constraint
        % Timing constraints for each job
        for i=1:n
            for t=1:A(i)-1
                X(t,i)==0;
            end
        end
    end
cvx_end

```

```

        for t=D(i)+1:T
            X(t,i)=0;
        end
    end
    sum(X)>=W';
cvx_end
theta = X./(s'*ones(1,n));
figure;
bar((s'*ones(1,n)).*theta,1,'stacked');
xlabel('Time: tt');
ylabel('Stacked speed: s_t');

```

Results:

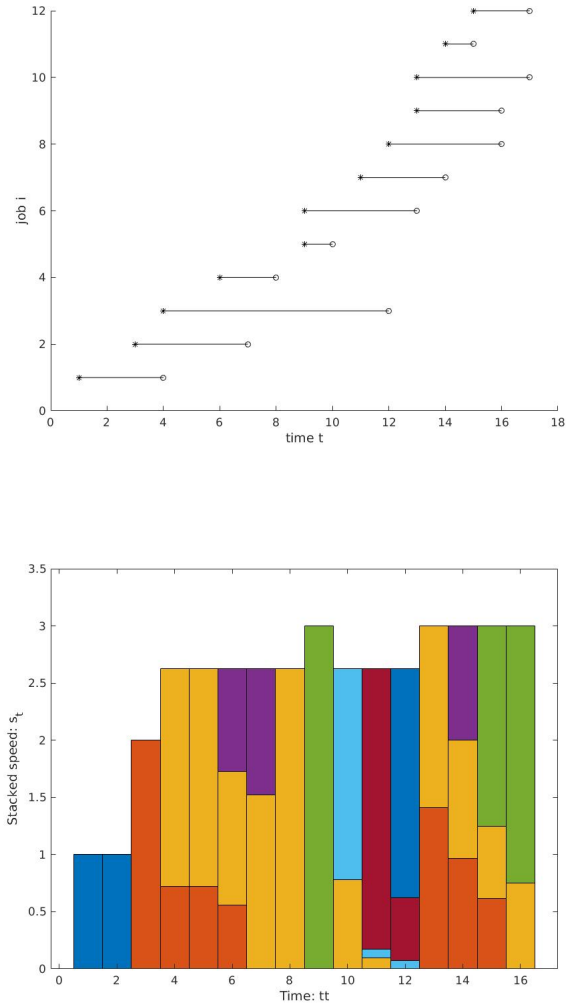


Figure 1: Stacked processor speed, each stack representing portion of speed devoted to allocated jobs.

17.4

We must maximize the net revenue minus the penalty to maximize the net profit:

$$\begin{aligned} & \text{maximize} && \text{trace}(R^T N) - p^T s \\ & \text{subject to} && s = q - A^T N T \\ & && N \succeq 0 \\ & && \sum_{t=1}^T N_{it} = I_i \end{aligned}$$

Code:

```
cvx_begin
    variable N(n,T)
    % we need s positive to ensure no negative penalties.
    s = pos(q-diag(Acontr'*N*Tcontr))
    maximize(sum(diag(R'*N())) - p'*s)
    subject to
        N >= 0;
        sum(N) == I';
cvx_end

net_profit = cvx_optval
revenue = sum(diag(R'*N))
payment = p'*s

% Highest ad revenue %
cvx_begin
    variable N(n,T)
    maximize (sum(diag(R'*N)))
    subject to
        N >= 0;
        ones(1,n)*N == I';
cvx_end
hi_ad_revenue = cvx_optval;
s = pos(q-diag(Acontr'*N*Tcontr))
hi_ad_payment = p'*s
hi_ad_net_profit = hi_ad_revenue-hi_ad_payment
```

Results:

Full ad portfolio:

net_profit = 230.5660

revenue = 268.2319

payment = 37.6659

Highest earning ad portfolio:
hi_ad_revenue = 305.1017
hi_ad_payment = 232.2602
hi_ad_net_profit = 72.8415

17.5

maximize
subject to

Code:

Results:

17.8

17.9