HW7

Carl Mueller
CSCI 5254 - Convex Optimization

April 26, 2018

8.16

Formulate the following asa 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 do 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 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:

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

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:

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

8.24

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

Proposition 1.

$$u^t x_i \leq ||u||_2 |x_i||_2 \tag{2}$$
$$||u||_2 ||x_i||_2 \leq p||x_i||_2 \tag{3}$$
$$u^t y_j \leq ||u||_2 |y_j||_2 \tag{4}$$
$$||u||_2 |y_j||_2 \leq p||y_j||_2 \tag{5}$$
$$-||u||_2 ||y_j||_2 \geq -p||y_j||_2 \tag{6}$$
$$\tag{7}$$

For x_i:

$$\begin{aligned} (a+u)^T x_i &\geq b \\ a^T x_i + u^T x_i &\geq b \\ a^T x_i + ||u||_2 |x_i||_2 &\geq b \\ a^T x_i + p|x_i||_2 &\geq b \\ a^T x_i - b &\geq -p|x_i||_2 \end{aligned}$$

For x_i:

$$\begin{aligned} (a+u)^T y_j &\leq b \\ a^T y_j + u^T y_j &\leq b \\ a^T y_j + ||u||_2 ||y_j||_2 &\leq b \\ a^T y_j + p||x_i||_2 &\leq b \\ a^T y_j - b &\leq -p||y_j||_2 \\ b - a^T y_j &\geq p||y_j||_2 \end{aligned}$$

The optimization problem:

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

Additional Exercises:

5.12

One heurisitc estiamte 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 $\hat{x}_\tau$ and $\hat{x}_{\tau-1}$ is below some tolerance, τ being the current iteration step.

Code:

```
tolerance = .00000000001
% Seed our initial estimate of x using huber function
cvx_begin
variable x(n);
    minimize( sum(huber(A*x-y)) );
cvx_end
x_prior = zeros(n)
while 1

    % Align the smallest indixes, find pi (the permutation index alignement)
    % 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
    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 estimated x (P=I) and x_true:"
norm(x_eye - x_true, 2)
"Distance x_true and estimated x:"
norm(x_true - x, 2)

miss_count = 0;
for i=1:size(A,1)
    if sum(P_true(i,:) ~= P_hat(i,:)) ~= 0
        miss_count = miss_count + 1;
    end
end
miss_count
```

```
\textbf{Results:}\\
Optimal value (cvx_optval): +441.802
"Distance estimated x (P=I) and x_true:"
ans = 3.4363
"Distance x_true and estimated x:"
ans = 0.0965
"Numer of mismatched rows in P_true vs P_hat:"
miss_count = 47
```

5.18

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

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

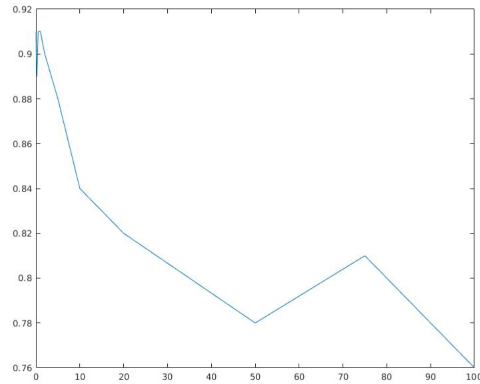


Figure 1: U-values vs percent correct.

This can be represented by the following problem:

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

This can be reexpressed using the individual inequality constraints:

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

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 predict predicted labels by computing the affine function
    % on xtest using the estimated optial 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
    percent_correct = correct/mTest
    E = [E ; percent_correct]
end
plot(U,E)
```

13.15

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

Proposition 1.

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

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

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

Proposition 2.

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

$$\begin{aligned}
E[(z-w^Tr)(z-w^Tr)] &\leq .01E[z^2] \\
E[z^Tz + r^Tww^Tr - 2zw^Tr] &\leq .01E[z^2] \\
E[z^Tz] + E[r^Tww^Tr] - 2E[zw^Tr] &\leq .01E[z^2] \\
E[z^Tz] + E[r^Tww^Tr] - 2E[zw^Tr] &\leq .01E[z^2] \\
E[z^Tz] + E[w^Trr^Tw] - 2E[(c^Tr)^Tw^Tr] &\leq .01E[z^2] \\
c^T diag(\Sigma) + \bar{r}^T cc^T \bar{r} + w^T(\Sigma + \bar{r}\bar{r}^T)w - 2E[(c^Tr)^Tw^Tr] &\leq .01E[z^2] \\
c^T diag(\Sigma) + \bar{r}^T cc^T \bar{r} + w^T(\Sigma + \bar{r}\bar{r}^T)w - 2w^T(\Sigma + \bar{r}\bar{r}^T)c &\leq .01E[z^2]
\end{aligned}$$

This becomes the optimization problem reflected in the matlab code:

$$\begin{array}{ll} \text{minimize} & ||w||_1 \\ \text{subject to} & c^T diag(\Sigma) + \bar{r}^T cc^T \bar{r} + w^T(\Sigma + \bar{r}\bar{r}^T)w - 2w^T(\Sigma + \bar{r}\bar{r}^T)c \leq .01E[z^2] \end{array}$$

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

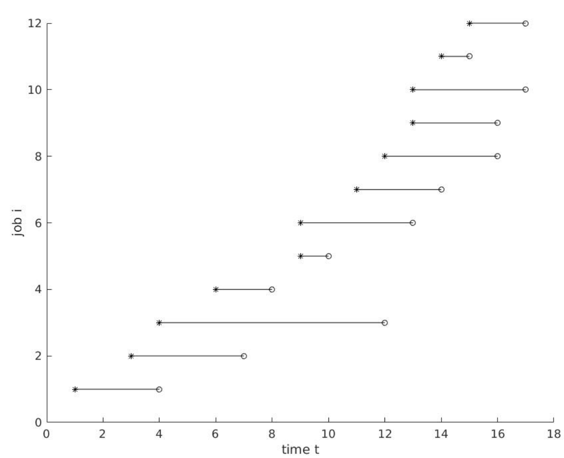
We begin by stating that $X = \theta s$. The optmization problem can be formulated as:

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

The top two constraitns are the processor speed limits and slew rates. Each row coefficients θ_{ti} for $1 = 1, \ldots, n$ 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
            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:



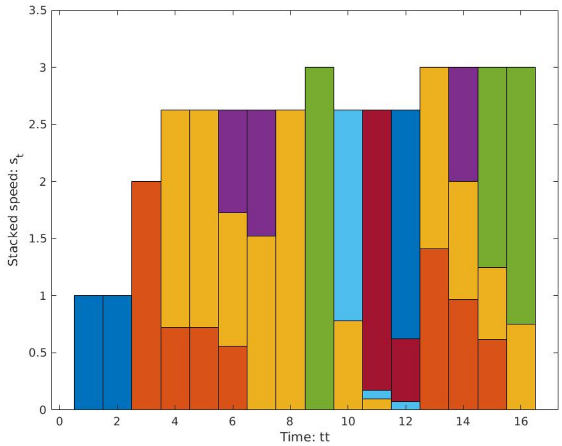


Figure 2: Stacked processor speed, each stack representing portion of speed devoted to allocated jobs.

17.4

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

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

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 porfolio:
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

For the first condition: $\phi(v) = v^2$ which will penalize higher values. For the second condition: $\phi(v) = huber(v)$ which is less sensitive to outliers and thus will allow a few large preference violations. The minimization problem for the squared error becomes becomes:

$$\begin{array}{ll} \text{minimize} & \sum \phi(V) \\ \text{subject to} & V = max(r_j + 1 - r_i, 0) \end{array}$$

For the huber penalty function, since CVX requires huber be passed an affine function, we must reformulate:

$$\begin{array}{ll} \text{minimize} & \sum \phi(V) \\ \text{subject to} & r_j + 1 - r_i, 0 \geq 0 \end{array}$$

Code:

```
cvx_begin
variable R(50)
V = max(R(preferences(:,2))+1-R(preferences(:,1)),0)
minimize(sum(square_phi(V)))
```

```
cvx_end
sum(V>0.001)
histogram(V)

cvx_begin
variable R(50)
V = R(preferences(:,2))+1-R(preferences(:,1))
minimize(sum(huber_phi(V)))
subject to
    R(preferences(:,2))+1-R(preferences(:,1)) >= 0
cvx_end
sum(V>0.001)
histogram(V)

function square_penalty = square_phi( x )
square_penalty = pow_pos(x, 2)
end

function huber_penalty = huber_phi( x )
huber_penalty = huber(x)
end
```

Results:
Squared penalty:
sum(V¿0.001) = 781
Huber Penalty:
sum(V¿0.001) = 900

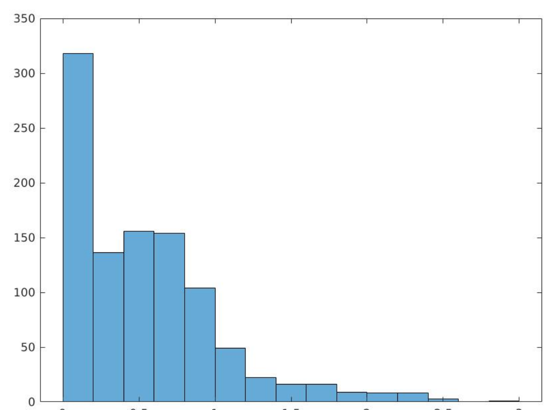


Figure 3: Histogram for square penalty.

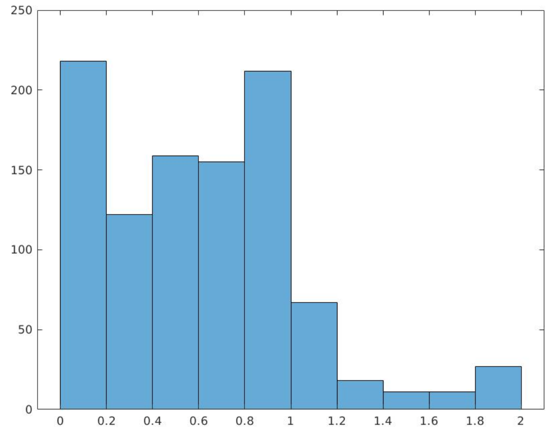


Figure 4: Histogram for huber penalty.

17.8

a)

Assume we have $P_1 = 10$ and $P_2 = 10$ where $K = 1, 2$. Let there be 2 positions and $w_1 = [1, 0]$ and $w_2 = [0, 1]$ and all $v = 0$ For quasiconcavity to hold, we need the following set to be concave $\{ x \mid V(x) \geq \alpha \}$. If we can find an example of a series of x in our domain that violates this then we've shows that the function violates the requiremenets for quasi convexity.

$$x_1 = [1, 0], x_2 = [0, 1], x_3 = [-1, 0], x_4 = [0, -1]$$

$$V(x_1) = 10 * \frac{1}{1 + e^{-[1,0]^T [1,0]}} + 10 * \frac{1}{1 + e^{-[0,1]^T [1,0]}} \approx 2.6 + 5 = 7.6$$

$$V(x_1) = 10 * \frac{1}{1 + e^{-[1,0]^T [0,1]}} + 10 * \frac{1}{1 + e^{-[0,1]^T [0,1]}} \approx 2.6 + 5 = 7.6$$

$$V(x_1) = 10 * \frac{1}{1 + e^{-[1,0]^T [-1,0]}} + 10 * \frac{1}{1 + e^{-[0,1]^T [-1,0]}} \approx 7.6 + 5 = 13.6$$

$$V(x_1) = 10 * \frac{1}{1 + e^{-[1,0]^T [-1,0]}} + 10 * \frac{1}{1 + e^{-[0,1]^T [0,-1]}} \approx 7.6 + 5 = 13.6$$

We can choose an choose an $\alpha = 10$ where despite all x being in some domain, the quasiconcavity set is violated.

b)

By restricting the above example with the constraint that $w_k^T x + v_k$, we ensure that each f is correctly refelcts political view w with with respect to the positions on X i.e. increases in position track the actual preferences of the voters. This means that for a given domain, there is an optimal.

c)

Code:

```
cvx_begin
variable x(n)
maximize(P'*gapx(W*x+v))
subject to
    W*x+v >= 0
     l <= x <= u
     l < 0
     u > 0
cvx_end
```

Results:

```
Increase in votes: +435580
 Political Position:
x = [1.0000, 1.0000, 0.1758, -0.5572, -1.0000]
```

17.9

The max function is a convex function, thus the objective is a linear comibnation of convex function which is itself convex.

a)

$$\begin{array}{ll} \text{minimize} & \sum_{j=1}^{J} w_j (t_j^{tar} - t + j)_+ \\ \text{subject to} & \sum_{p=1}^{P} x_p \preceq x^{tot} \\ & l^{job} \preceq l^{max} \end{array}$$

b)

Code:

```
cvx_begin
variable t(J)
variable x(n,P)
minimize(sum(w'*max(t_tar-t, zeros(J,1))))
subject to
```

```
    t_tar-t >= zeros(J,1)
    t >= 0
    x >= x_min
    R'*(pow_p(diag(A'*x) - R*t, -1)) <= l_max;
    size(R*t)
    diag(A'*x) > R*t;
    sum(x,2) <= x_tot
cvx_end
t
t_tar
```

Results:

```
Optimal value (cvx_optval): +7.74196

t =
    9.0300
    8.5631
    6.8426
    1.6032
    6.3854
    9.9212
    9.3040
    5.9270
    2.0905
    2.8096

t_tar =
    9.0300
    8.5631
    6.8426
    8.6306
    6.3854
    9.9212
    9.3040
    5.9270
    2.0905
```