HW V Solutions

EE 588: Optimization for the Information and Data Sciences

University of Southern California

Estimating a vector with unknown measurement nonlinearity. (A specific instance of exercise 7.9 in *Convex Optimization*.) We want to estimate a vector $x \in \mathbf{R}^n$, given some measurements

$$y_i = \phi(a_i^T x + v_i), \quad i = 1, \ldots, m.$$

Here $a_i \in \mathbf{R}^n$ are known, v_i are IID $\mathcal{N}(0, \sigma^2)$ random noises, and $\phi : \mathbf{R} \to \mathbf{R}$ is an unknown monotonic increasing function, known to satisfy

$$\alpha \leq \phi'(u) \leq \beta,$$

for all u. (Here α and β are known positive constants, with $\alpha < \beta$.) We want to find a maximum likelihood estimate of x and ϕ, given y_i. (We also know a_i, σ, α, and β.)

This sounds like an infinite-dimensional problem, since one of the parameters we are estimating is a function. In fact, we only need to know the m numbers $z_i = \phi^{-1}(y_i)$, $i = 1, \ldots, m$. So by estimating ϕ we really mean estimating the m numbers $z_1, \ldots, z_m$. (These numbers are not arbitrary; they must be consistent with the prior information $\alpha \leq \phi'(u) \leq \beta$ for all u.)

(a) Explain how to find a maximum likelihood estimate of x and ϕ (*i.e.*, $z_1, \ldots, z_m$) using convex optimization.

(b) Carry out your method on the data given in `nonlin_meas_data.m`, which includes a matrix $A \in \mathbf{R}^{m \times n}$, with rows $a_1^T, \ldots, a_m^T$. Give $\hat{x}_{\mathrm{ml}}$, the maximum likelihood estimate of x. Plot your estimated function $\hat{\phi}_{\mathrm{ml}}$. (You can do this by plotting $(\hat{z}_{\mathrm{ml}})_i$ versus y_i, with y_i on the vertical axis and $(\hat{z}_{\mathrm{ml}})_i$ on the horizontal axis.)

Hint. You can assume the measurements are numbered so that y_i are sorted in nondecreasing order, *i.e.*, $y_1 \leq y_2 \leq \cdots \leq y_m$. (The data given in the problem instance for part (b) is given in this order.)

Solution.

(a) We can write the measurement model as

$$\phi^{-1}(y_i) = a_i^T x + v_i, \quad i = 1, \ldots, m.$$

The function ϕ^{-1} is unknown, but it has derivatives between $1/\beta$ and $1/\alpha$. Therefore $z_i = \phi^{-1}(y_i)$ and y_i must satisfy the inequalities

$$\frac{y_{i+1} - y_i}{\beta} \leq z_{i+1} - z_i \leq \frac{y_{i+1} - y_i}{\alpha}, \qquad i = 1, \ldots, m-1,$$

if we assume that the points are sorted with y_i in increasing order. Conversely, if z and y satisfy these inequalities, then there exists a nonlinear function ϕ with $y_i = \phi(z_i)$, $i = 1, \ldots, m$, and with derivatives between α and β (for example, a piecewise-linear function that interpolates the points). Therefore, as suggested in the problem statement, we can use $z_1, \ldots, z_m$ as parameters instead of ϕ.

The log-likelihood function is

$$l(z, x) = -\frac{1}{2\sigma^2}\sum_{i=1}^{m}(z_i - a_i^T x)^2 - m\log(\sigma\sqrt{2\pi}).$$

Thus to find a maximum likelihood estimate of x and z one solves the problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{m}(z_i - a_i^T x)^2 \\ \text{subject to} & (y_{i+1} - y_i)/\beta \leq z_{i+1} - z_i \leq (y_{i+1} - y_i)/\alpha, \quad i = 1, \ldots, m-1. \end{array}$$

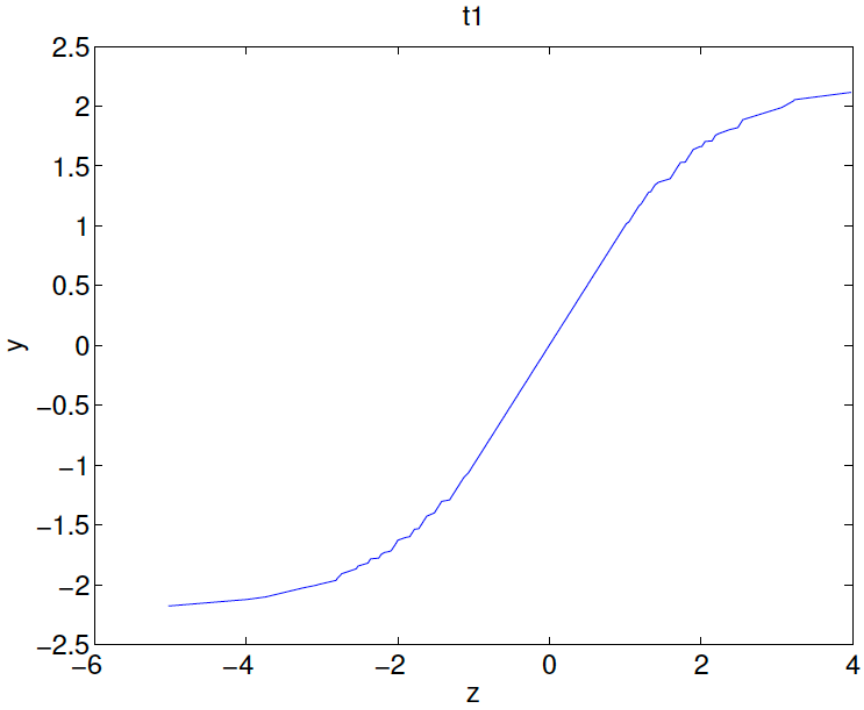
This is a quadratic program with variables $z \in \mathbf{R}^m$ and $x \in \mathbf{R}^n$.

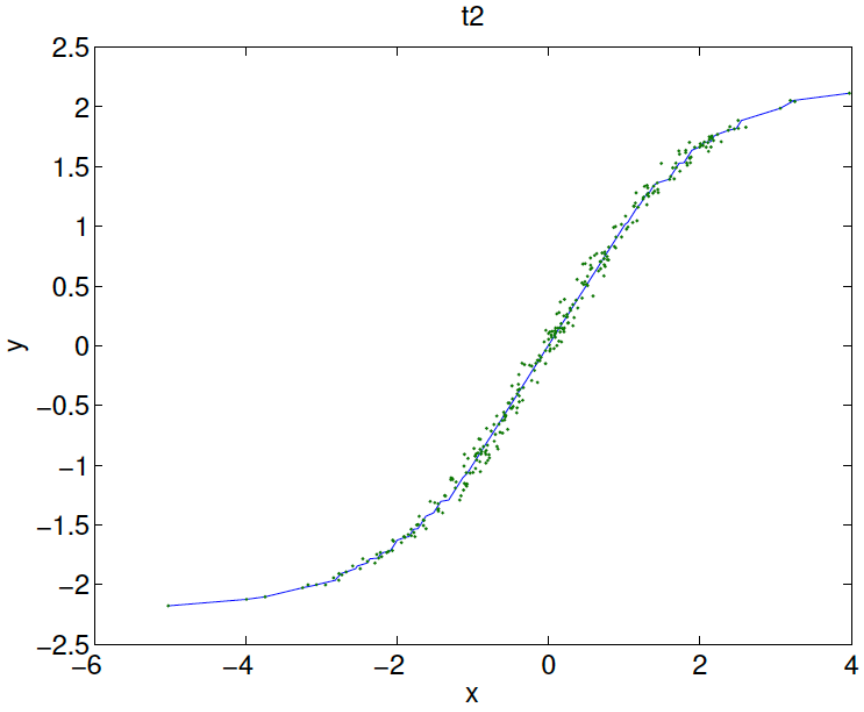
(b) The following Matlab code solve the given problem

```
nonlin_meas_data

row=zeros(1,m);
row(1)=-1;
row(2)=1;
col=zeros(1,m-1);
col(1)=-1;
B=toeplitz(col,row);

cvx_begin
    variable x(n);
    variable z(m);
    minimize(norm(z-A*x));
    subject to
       1/beta*B*y<=B*z;
       B*z<=1/alpha*B*y;
cvx_end

disp('estimated x:'); disp(x);

plot(z,y)
ylabel('y')
xlabel('z')
title('ML estimate of \phi')
```

The estimated x is $x = (0.4819, \ -0.4657, \ 0.9364, \ 0.9297)$.

The first figure shows the estimate function ϕ. The second figure shows ϕ and the data points $a_i^T x$, y_i.


t1
2.5
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2.5
y
−6
−4
−2
0
2
4
z


t2
2.5
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2.5
y
−6
−4
−2
0
2
4
x

Maximum likelihood estimation of an increasing nonnegative signal. We wish to estimate a scalar signal $x(t)$, for $t = 1, 2, \ldots, N$, which is known to be nonnegative and monotonically nondecreasing:

$$0 \leq x(1) \leq x(2) \leq \cdots \leq x(N).$$

This occurs in many practical problems. For example, $x(t)$ might be a measure of wear or deterioration, that can only get worse, or stay the same, as time t increases. We are also given that $x(t) = 0$ for $t \leq 0$.

We are given a noise-corrupted moving average of x, given by

$$y(t) = \sum_{\tau=1}^{k} h(\tau)x(t-\tau) + v(t), \quad t = 2, \ldots, N+1,$$

where $v(t)$ are independent $\mathcal{N}(0,1)$ random variables.

(a) Show how to formulate the problem of finding the maximum likelihood estimate of x, given y, taking into account the prior assumption that x is nonnegative and monotonically nondecreasing, as a convex optimization problem. Be sure to indicate what the problem variables are, and what the problem data are.

(b) We now consider a specific instance of the problem, with problem data (*i.e.*, N, k, h, and y) given in the file `ml_estim_incr_signal_data.*`. (This file contains the true signal `xtrue`, which of course you cannot use in creating your estimate.) Find the maximum likelihood estimate $\hat{x}_{\mathrm{ml}}$, and plot it, along with the true signal. Also find and plot the maximum likelihood estimate $\hat{x}_{\mathrm{ml,free}}$ *not taking into account the signal nonnegativity and monotonicity.*
Hints.

- Matlab: The function `conv` (convolution) is overloaded to work with CVX.
- Python: Numpy has a function `convolve` which performs convolution. CVXPY has `conv` which does the same thing for variables.
- Julia: The function `conv` is overloaded to work with Convex.jl.

Solution.

(a) To simplify our notation, we let the signal $\hat{y}_x$ be the noiseless moving average of the signal x. That is,

$$\hat{y}_x(t) = \sum_{\tau=1}^{k} h(\tau)x(t-\tau), \quad t = 2, \ldots, N+1.$$

Note that $\hat{y}_x$ is a linear function of x.
The nonnegativity and monotonicity constraint on x can be expressed as a set of linear inequalities,

$$x(1) \geq 0, \quad x(1) \leq x(2), \quad \ldots \quad x(N-1) \leq x(N).$$

Now we turn to the maximum likelihood problem. The likelihood function is

$$\prod_{t=2}^{N+1} p\left(y(t) - \hat{y}_x(t)\right),$$

where p is the density function of a $\mathcal{N}(0,1)$ random variable. The negative log-likelihood function has the form

$$\alpha + \beta \|\hat{y}_x - y\|_2^2,$$

where α is a constant, and β is a positive constant. Thus, the ML estimate is found by minimizing the quadratic objective $\|\hat{y_x} - y\|_2^2$. The ML estimate when we *do not* take into account signal nonnegativity and monotonicity can be found by solving a least-squares problem,

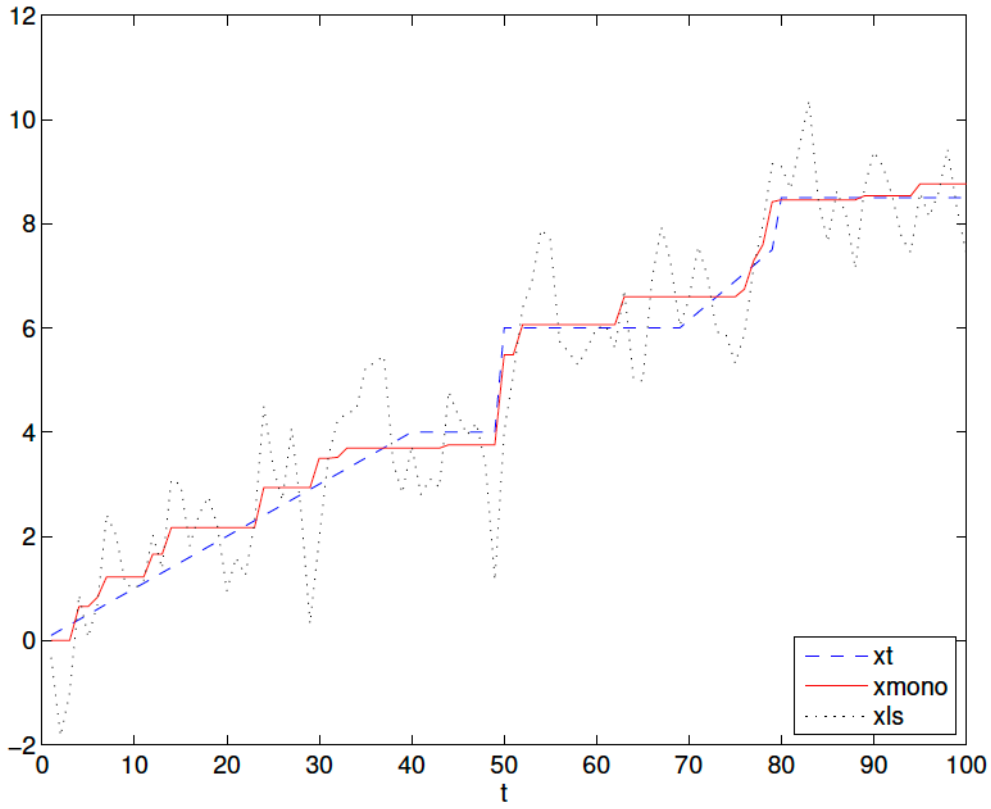
$$\hat{x}_{\text{ml,free}} = \operatorname*{argmin}_x \|\hat{y}_x - y\|_2^2.$$

Since convolution is an invertible linear operation, to get the ML estimate without the monotonicity constraint, we simply apply deconvolution. Since the number of measurements and variables to estimate are the same (N), there is no smoothing effect to reduce noise, and we can expect the deconvolved estimate to be poor.

With the prior assumption that the signal x is nonnegative and monotonically nondecreasing we can find the ML estimate $\hat{x}_{\text{ml}}$ by solving the QP

$$\begin{array}{ll} \text{minimize} & \|\hat{y}_x - y\|_2^2 \\ \text{subject to} & x(1) \geq 0 \\ & x(t) \leq x(t+1) \quad t = 1, \ldots, N-1. \end{array}$$

(b) The ML estimate for the given problem instance, with and without the assumption of nonnegativity and monotonicity, is plotted below. We've also plotted the true signal x. We observe that the ML estimate $\hat{x}_{\text{ml}}$, which takes into consideration nonnegativity and monotonicity, is a much better estimate of signal x than the simple unconstrained solution $\hat{x}_{\text{ml,free}}$ obtained via deconvolution.



Plots for the other languages are shown below.

The following Matlab code computes the ML solutions for the constrained and unconstrained estimation problems.

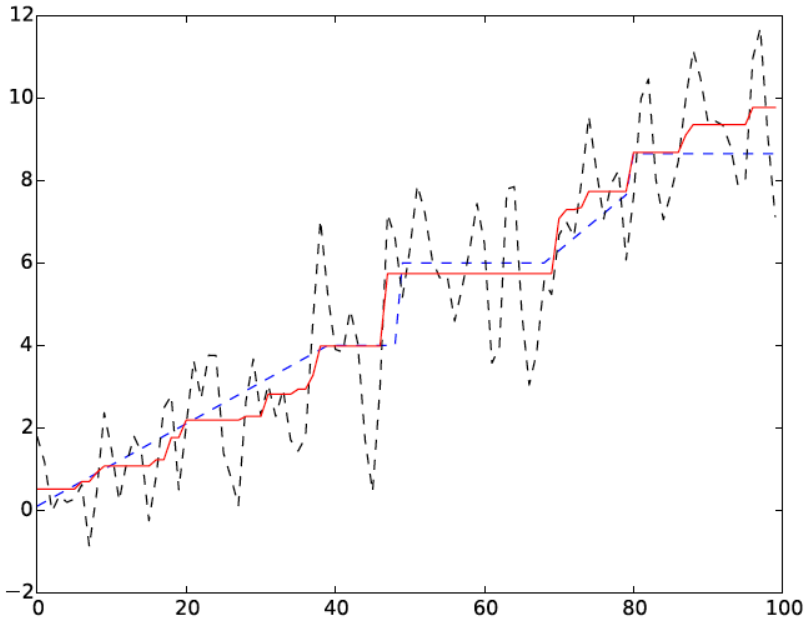
```
% ML estimation of increasing nonnegative signal
% problem data
ml_estim_incr_signal_data;

% maximum likelihood estimation with no monotonicity taken in to account
% can be solved analytically
cvx_begin
    variable xls(N)
    yhat = conv(h,xls);    % estimated output
    % yhat is truncated to match problem description
    minimize (sum_square(yhat(1:end-3) - y))
cvx_end

% monotonic and non-negative signal estimation
cvx_begin
```

```
    variable xmono(N)
    yhat = conv(h,xmono);  % estimated output
    minimize (sum_square(yhat(1:end-3) - y))
    subject to
        xmono(1) >= 0;
        xmono(1:N-1) <= xmono(2:N);
cvx_end

t = 1:N;
figure; set(gca, 'FontSize',12);
plot(t,xtrue,'--',t,xmono,'r',t,xls,'k:');
xlabel('t'); legend('xt','xmono','xls','Location','SouthEast');
%print -depsc ml_estim_incr_signal_plot
```

The following code implements the same solution in Python

```
import cvxpy as cvx
import matplotlib.pyplot as plt

# Load data file, gives us N and Y
from ml_estim_incr_signal_data import *

# Estimate signal without constraints
xls = cvx.Variable(N)
yhat = cvx.conv(h,xls)
error = cvx.sum_squares(yhat[0:-3] - y)
prob = cvx.Problem(cvx.Minimize(error))
prob.solve()

xmono = cvx.Variable(N)
yhat = cvx.conv(h,xmono)
error = cvx.sum_squares(yhat[0:-3] - y)
constraints = [xmono[0] >= 0]
constraints += [xmono[0:-1] <= xmono[1:]]
prob2 = cvx.Problem(cvx.Minimize(error),constraints)
prob2.solve()

fig = plt.figure()
t = np.arange(N)
plt.plot(t,xtrue,'b--',t,xls.value,'k--',t,xmono.value,'r')
plt.show()
plt.savefig('ml_estim_incr_signal_plot.eps',format='eps')
```

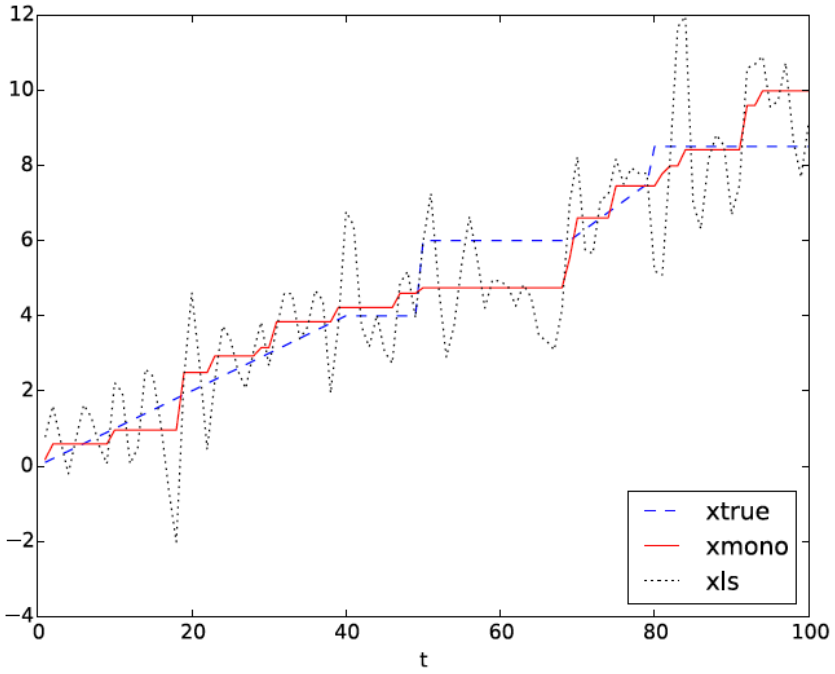
The plot below shows the results from the Python implementation.



In Julia, the code is written as follows.

```
# ML estimation of increasing nonnegative signal
# problem data
include("ml_estim_incr_signal_data.jl");

using Convex

# maximum likelihood estimation with no monotonicity taken in to account
# can be solved analytically
xls = Variable(N);
yhat = conv(h, xls); # estimated output
# yhat is truncated to match problem description
p = minimize (sum_squares(yhat[1:end-3] - y));
solve!(p);

# monotonic and non-negative signal estimation
xmono = Variable(N);
yhat = conv(h, xmono);  # estimated output
constraints = [xmono[1] >= 0, xmono[1:N-1] <= xmono[2:N]]
p = minimize (sum_squares(yhat[1:end-3] - y), constraints);
solve!(p);
```

```
using PyPlot
t = 1:N;
figure();
hold(true);
plot(t, xtrue, "b--", label="xtrue");
plot(t, xmono.value, "r", label="xmono");
plot(t, xls.value, "k:", label="xls");
xlabel("t");
legend(loc=4);
savefig("ml_estim_incr_signal.eps");
```



Three-way linear classification. We are given data

$$x^{(1)}, \ldots, x^{(N)}, \quad y^{(1)}, \ldots, y^{(M)}, \quad z^{(1)}, \ldots, z^{(P)},$$

three nonempty sets of vectors in $\mathbf{R}^n$. We wish to find three affine functions on $\mathbf{R}^n$,

$$f_i(z) = a_i^T z - b_i, \quad i = 1, 2, 3,$$

that satisfy the following properties:

$$\begin{array}{l} f_1(x^{(j)}) > \max\{f_2(x^{(j)}), f_3(x^{(j)})\}, \quad j = 1, \ldots, N, \\ f_2(y^{(j)}) > \max\{f_1(y^{(j)}), f_3(y^{(j)})\}, \quad j = 1, \ldots, M, \\ f_3(z^{(j)}) > \max\{f_1(z^{(j)}), f_2(z^{(j)})\}, \quad j = 1, \ldots, P. \end{array}$$

In words: f_1 is the largest of the three functions on the x data points, f_2 is the largest of the three functions on the y data points, f_3 is the largest of the three functions on the z data points. We can give a simple geometric interpretation: The functions f_1, f_2, and f_3 partition $\mathbf{R}^n$ into three regions,

$$\begin{array}{rcl} R_1 & = & \{z \mid f_1(z) > \max\{f_2(z), f_3(z)\}\}, \\ R_2 & = & \{z \mid f_2(z) > \max\{f_1(z), f_3(z)\}\}, \\ R_3 & = & \{z \mid f_3(z) > \max\{f_1(z), f_2(z)\}\}, \end{array}$$

defined by where each function is the largest of the three. Our goal is to find functions with $x^{(j)} \in R_1$, $y^{(j)} \in R_2$, and $z^{(j)} \in R_3$.

Pose this as a convex optimization problem. You may not use strict inequalities in your formulation.

Solve the specific instance of the 3-way separation problem given in `sep3way_data.m`, with the columns of the matrices `X`, `Y` and `Z` giving the $x^{(j)}$, $j = 1, \ldots, N$, $y^{(j)}$, $j = 1, \ldots, M$ and $z^{(j)}$, $j = 1, \ldots, P$. To save you the trouble of plotting data points and separation boundaries, we have included the plotting code in `sep3way_data.m`. (Note that `a1`, `a2`, `a3`, `b1` and `b2` contain arbitrary numbers; you should compute the correct values using CVX.)

Solution. The inequalities

$$\begin{array}{l} f_1(x^{(j)}) > \max\{f_2(x^{(j)}), f_3(x^{(j)})\}, \quad j = 1, \ldots, N, \\ f_2(y^{(j)}) > \max\{f_1(y^{(j)}), f_3(y^{(j)})\}, \quad j = 1, \ldots, M, \\ f_3(z^{(j)}) > \max\{f_1(z^{(j)}), f_2(z^{(j)})\}, \quad j = 1, \ldots, P. \end{array}$$

are homogeneous in a_i and b_i so we can express them as

$$\begin{array}{l} f_1(x^{(j)}) \geq \max\{f_2(x^{(j)}), f_3(x^{(j)})\} + 1, \quad j = 1, \ldots, N, \\ f_2(y^{(j)}) \geq \max\{f_1(y^{(j)}), f_3(y^{(j)})\} + 1, \quad j = 1, \ldots, M, \\ f_3(z^{(j)}) \geq \max\{f_1(z^{(j)}), f_2(z^{(j)})\} + 1, \quad j = 1, \ldots, P. \end{array}$$

Note that we can add any vector α to each of the a_i, without affecting these inequalities (which only refer to difference between a_i's), and we can add any number β to each of the b_i's for the same reason. We can use this observation to normalize or simplify the a_i and b_i. For example, we can assume without loss of generality that $a_1 + a_2 + a_3 = 0$ and $b_1 + b_2 + b_3 = 0$.

The following script implements this method for 3-way classification and tests it on a small separable data set

```
clear all; close all;
% data for problem instance
M = 20;
N = 20;
P = 20;

X = [

    3.5674    4.1253    2.8535    5.1892    4.3273    3.8133    3.4117 ...
    3.8636    5.0668    3.9044    4.2944    4.7143    3.3082    5.2540 ...
    2.5590    3.6001    4.8156    5.2902    5.1908    3.9802 ;...
   -2.9981    0.5178    2.1436   -0.0677    0.3144    1.3064    3.9297 ...
    0.2051    0.1067   -1.4982   -2.4051    2.9224    1.5444   -2.8687 ...
    1.0281    1.2420    1.2814    1.2035   -2.1644   -0.2821];

Y = [
   -4.5665   -3.6904   -3.2881   -1.6491   -5.4731   -3.6170   -1.1876 ...
   -1.0539   -1.3915   -2.0312   -1.9999   -0.2480   -1.3149   -0.8305 ...
   -1.9355   -1.0898   -2.6040   -4.3602   -1.8105    0.3096; ...
    2.4117    4.2642    2.8460    0.5250    1.9053    2.9831    4.7079 ...
    0.9702    0.3854    1.9228    1.4914   -0.9984    3.4330    2.9246 ...
    3.0833    1.5910    1.5266    1.6256    2.5037    1.4384];

Z = [
    1.7451    2.6345    0.5937   -2.8217    3.0304    1.0917   -1.7793 ...
    1.2422    2.1873   -2.3008   -3.3258    2.7617    0.9166    0.0601 ...
   -2.6520   -3.3205    4.1229   -3.4085   -3.1594   -0.7311; ...
   -3.2010   -4.9921   -3.7621   -4.7420   -4.1315   -3.9120   -4.5596 ...
   -4.9499   -3.4310   -4.2656   -6.2023   -4.5186   -3.7659   -5.0039 ...
   -4.3744   -5.0559   -3.9443   -4.0412   -5.3493   -3.0465];

cvx_begin
variables a1(2) a2(2) a3(2) b1 b2 b3
    a1'*X-b1 >= max(a2'*X-b2,a3'*X-b3)+1;
    a2'*Y-b2 >= max(a1'*Y-b1,a3'*Y-b3)+1;
    a3'*Z-b3 >= max(a1'*Z-b1,a2'*Z-b2)+1;
    a1 + a2 + a3 == 0
    b1 + b2 + b3 == 0
cvx_end

% now let's plot the three-way separation induced by
% a1,a2,a3,b1,b2,b3
% find maximally confusing point
p = [(a1-a2)';(a1-a3)']\[(b1-b2);(b1-b3)];
```

```
% now let's plot the three-way separation induced by
% a1,a2,a3,b1,b2,b3
% find maximally confusing point
p = [(a1-a2)';(a1-a3)']\[(b1-b2);(b1-b3)];

% plot
t = [-7:0.01:7];
u1 = a1-a2; u2 = a2-a3; u3 = a3-a1;
v1 = b1-b2; v2 = b2-b3; v3 = b3-b1;
line1 = (-t*u1(1)+v1)/u1(2); idx1 = find(u2'*[t;line1]-v2>0);
line2 = (-t*u2(1)+v2)/u2(2); idx2 = find(u3'*[t;line2]-v3>0);
line3 = (-t*u3(1)+v3)/u3(2); idx3 = find(u1'*[t;line3]-v1>0);
plot(X(1,:),X(2,:),'*',Y(1,:),Y(2,:),'ro',Z(1,:),Z(2,:),'g+',...
     t(idx1),line1(idx1),'k',t(idx2),line2(idx2),'k',t(idx3),line3(idx3),'k');


axis([-7 7 -7 7]);
```

The following figure is generated.