CSCI5254

Homework Set #7

[Boyd&Vandenberghe, Convex Optimization]:

Exercises 8.16, 8.24

[Additional Exercises,
http://spot.colorado.edu/~lich1539/cvx/extra_exercises.pdf **]**

Exercises 5.12, 5.18, 13.15, 16.5, 17.4, 17.5, 17.4, 17.8, 17.9

Matlab file for data for exercise 5.12:

```
% ls_perm_meas_data.m

rand('seed',0);randn('seed',0)
m=100;
k=40; % max # permuted measurements
n=20;
A=10*randn(m,n);
x_true=randn(n,1); % true x value
y_true=A*x_true + randn(m,1);
% build permuted indices
perm_idxs=randperm(m);perm_idxs=sort(perm_idxs(1:k));
new_pos=perm_idxs(randperm(k));
% true permutation matrix
P=eye(m);P(perm_idxs,:)=P(new_pos,:);
perm_idxs(find(perm_idxs==new_pos))=[];
y=P*y_true;
clear new_pos
```

Matlab file for data for exercise 5.18:

```
% data file for multi-label SVM problem
clear all;
randn('state', 0);
mTrain = 1000; % size of training data
mTest  = 100;  % size of test data
K = 10;        % number of categories
n = 20;        % number of features
A_true = randn(K, n);
b_true = randn(K, 1);
v = 0.2*randn(K, mTrain + mTest); % noise
data = randn(n, mTrain + mTest);
[~, label] = max(A_true * data + b_true * ones(1,
mTrain + mTest) + v, [], 1);
```

```
% training data
x = data(:, 1:mTrain);
y = label(1:mTrain);
% test data
xtest = data(:, (mTrain+1):end);
ytest = label((mTrain+1):end);
```

Matlab file for data for exercise 13.15:

```
% sparse_idx_track_data.m
% data generation for sparse index tracking problem
% generates n, c, rbar, Sigma

rand('state',1);
randn('state',1);

n = 500; % number of stocks
k = 20;  % number of factors (not described or needed
in problem statement)

% generate rbar, Sigma
F = 1.5*rand(n,k)-0.5; % factor matrix, entries uniform
on [-0.5,1]
Sigma = 0.5*F*diag(rand(k,1))*F' + diag(0.1*rand(n,1));
mu_rf = 0.2;                          % risk-free
return (weekly, 5% annual)
SR = 0.4;                             % Sharpe ratio
rbar = mu_rf + SR*sqrt(diag(Sigma));  % expected return
c = 5 + exp(2*randn(n,1));            % market
capitalization (index weights)
```

Matlab file for data for exercise 16.5:

```
% proc_sched_data.m
% Data for minimum energy processor speed scheduling.

n = 12;  % number of jobs.
T = 16;  % number of time periods.

Smin = 1;  % min processor speed.
Smax = 4;  % max processor speed.
R = 1;  % max slew rate.
```

```
% Parameters in power/speed model.
alpha = 1;
beta = 1;
gamma = 1;

% Job arrival times and deadlines.
A = [1; 3;  4; 6; 9;  9; 11; 12; 13; 13; 14; 15];
D = [3; 6; 11; 7; 9; 12; 13; 15; 15; 16; 14; 16];
% Total work for each job.
W = [2; 4; 10; 2; 3; 2; 3; 2; 3; 4; 1; 4];

% Plot showing job availability times & deadlines.
figure;
hold on;
scatter(A,[1:n],'k*');
scatter(D+1,[1:n],'ko');
for i=1:n
    plot([A(i),D(i)+1],[i,i],'k-');
end
hold off;
xlabel('time t');
ylabel('job i');
```

Matlab file for data for exercise 17.4:

```
%  ad_disp_data.m
%  data for online ad display problem
rand('state',0);
n=100;      %number of ads
m=30;       %number of contracts
T=60;       %number of periods

I=10*rand(T,1);  %number of impressions in each period
R=rand(n,T);    %revenue rate for each period and ad
q=T/n*50*rand(m,1);     %contract target number of impressions
p=rand(m,1);  %penalty rate for shortfall
Tcontr=(rand(T,m)>.8); %one column per contract. 1's at the periods to be displayed
for i=1:n
      contract=ceil(m*rand);
      Acontr(i,contract)=1; %one column per contract. 1's at the ads to be displayed
end
```

Matlab File for data for exercise 17.5:
http://spot.colorado.edu/~lich1539/cvx/rank_aggr_data.m

Matlab file for data for exercise 17.8:

```
% opt_pol_pos_data.m
% optimal political positioning.


% Set random seeds
randn('state',0);
rand('state',0);

% Set parameters
K = 12; % demographic groups (constituencies)
P = 20000 + ceil(50000*rand(K,1)); % group populations
n = 5; % political issues
W = randn(K,n); % W(k,:) = (w_k)^T political views of each constituency
v = rand(K,1); % v(k) = v_k preference of constituency for prior position
l= -ones(n,1); % lower limit on position change
u= ones(n,1); % upper limit on position change

g = @(z) 1./(1+exp(-z)); % logistic function
gp = @(z) exp(-z)./(1+exp(-z)).^2; % derivative of logistic function
gapxi = @(z,i) g(i)+gp(i)*(z-i);
% concave approximation to logistic function (upper bound when z >= 0) at point i
gapx = @(z) min(min(min(min(min(1,gapxi(z,0)),gapxi(z,1)),gapxi(z,2)),gapxi(z,3)),gapxi(z,4));
% concave approximation to logistic function (upper bound when z >= 0)
```

Matlab File for data for exercise 17.9:

```
% resource allocation for stream processing.

rand('state',1);
randn('state',1);

J = 10; % number of job types
P = 20; % number of process types
n = 5; % number of system resources

R = sprandn(P,J,0.6);
R = abs(R) > 0;  % job process matrix

x_tot = 10*(1+rand(n,1)); % total resources available
A = 3+7*rand(n,P);
x_min = 0.2*rand(n,P);  % minimum resources to run each
process
w = 1+rand(J,1);
t_tar = 2+8*rand(J,1);  % target job traffic
l_max = 1+2*rand(J,1); % maximum allowed job latency
```