```
%-------------------------------------------------------------
% Tony Hyun Kim
% CS 246, PS 4, Problem 1(a)
% Load trades
%-------------------------------------------------------------
clear all; close all;

%part a)
%plot all the trades
tradeFolder='Trades/DATA/';
quoteFolder='Quotes/DATA/';
daten=05;

files = dir(strcat(tradeFolder,'679.asc'));
for i = 1:size(files,1)
    f = files(i);
    name = strcat(tradeFolder, f.name );
    fid = fopen(name);
    C = textscan(fid, '%s %s %f %d  %*[^\n]', 'delimiter', ',', ...
             'EmptyValue', -Inf);
    fclose(fid);
    time=char(C{2});
    date=char(C{1});
    dateidx=(str2num(date(:,4:5))==daten); %#ok<*ST2NM>

    hour=str2num(time(dateidx,1:2));
    minute=str2num(time(dateidx,4:5));
    sec=str2num(time(dateidx,7:end));
    tsecs=hour*60*60+minute*60+sec;

    price=C{3}(dateidx);

    name = strcat(quoteFolder, f.name );
    fid = fopen(name);
    C = textscan(fid, '%s %s %s %f %f %d %d  %*[^\n]',...
       'delimiter', ',','EmptyValue', -Inf);
    fclose(fid);

    date=char(C{1});
    dateidx=(str2num(date(:,4:5))==daten);

    time=char(C{2});
    exch=char(C{3});
    idx=(exch=='N') & dateidx;

    hour=str2num(time(idx,1:2));
    minute=str2num(time(idx,4:5));
    sec=str2num(time(idx,7:end));
    qsecs=hour*60*60+minute*60+sec;
    bid=C{4}(idx);
    ask=C{5}(idx);

    figure;
    starttime=10*60*60; %10am
    pidx=(tsecs>starttime) & (tsecs<(starttime+6*60*60)); %plus 6h
    qidx=(qsecs>starttime) & (qsecs<(starttime+6*60*60)); %plus 6h

    plot(tsecs(pidx),price(pidx),'b',qsecs(qidx),bid(qidx),'r',...
    qsecs(qidx),ask(qidx),'g');
    hold on;
    legend('tradeprice','bid','ask')
end

close all;
q1a_removebad;
```

```
%-------------------------------------------------------------
% Tony Hyun Kim
% CS 246, PS 4, Problem 1(a)
% Remove bad trades
%-------------------------------------------------------------
t_in = tsecs(pidx);
p_in = price(pidx);

q_in = qsecs(qidx);
b_in = bid(qidx);
a_in = ask(qidx);

[q_in ind] = unique(q_in);
b_in = b_in(ind);
a_in = a_in(ind);

plot(t_in,p_in); hold on;
plot(q_in,b_in,'r');

t = [];
p = [];

for i = 1:length(t_in)
    % Current value
    t0 = t_in(i);
    p0 = p_in(i);

    % Interpolate bid and ask
    bi = interp1(q_in,b_in,t_in(i));
    ai = interp1(q_in,a_in,t_in(i));

    if((bi<=p0)&&(p0<=ai)) % Definition of good trade
        t = [t; t0];
        p = [p; p0];
    end
end

grid on;
set(gca,'FontSize',16);
plot(t,p,'Linewidth',2);
xlabel('Time (s)');
ylabel('Price');
% title('Stock #679');
```

```
%-------------------------------------------------------------
% Tony Hyun Kim
% CS 246, PS 4, Problem 1(b)
% Stream through price and compute rolling statistics
%   and make trades
%-------------------------------------------------------------

% Previously, we computed the trade time and price
%   in the vectors t and p respectively

% Training time is 10 AM to 3 PM, i.e. 5 total hours. Note
%   that t(1) has been set in 'load_file.m' to be 10 AM.
t_tr = t(1) + 5*60*60;
N = find(t_tr<t,1); % Training examples

X = zeros(N,2); % Format: [Small Big] with sign for up/down
Y = zeros(N,1);

% We shall stream through the N data points, and implement
%   the shift register and rolling queue
shift  = zeros(2,4); % shift(1/2,:) = min/max
pqueue = zeros(2,1); % pqueue(1/2)  = min/max

minute_prev = -1;
for i = 2:(N-1)
    % Data structure for cached rolling min-high
    %-------------------------------------------------
    minute = floor(t(i)/60);
    if(minute==minute_prev) % Still in the same minute
        % Update priority queue
        if(p(i)>pqueue(2))  % Current price larger than max
            pqueue(2) = p(i);
        elseif(p(i)<pqueue(1)) % Smaller than min
            pqueue(1) = p(i);
        end
    else % Minute is up
        % Load into priority queue (at front)
        shift = [pqueue shift(:,1:(end-1))];

        % Reset priority queue with new value
        pqueue = p(i)*[1 1]';
    end
    minute_prev = minute;

    % Compute rolling high-low
    %-------------------------------------------------
    high = max([shift(2,:) pqueue(2)]);
    low  = min([shift(1,:) pqueue(1)]);
    highlow = high-low;

    % Classify ticks among
    %   {up-small, down-small, up-big, down-big}
    %-------------------------------------------------
    thresh   = 0.5*highlow;
    prevmove = p(i)-p(i-1);
    if(abs(prevmove)>thresh) % Big move
        X(i,2) = sign(prevmove);
    else % Small move
        X(i,1) = sign(prevmove);
    end
    Y(i) = (p(i+1)-p(i))/p(i); % Percentage change of next move
end

% Statistics of regression
%-------------------------------------------------

% Remove sparse entries in X/Y (i.e. no price move recorded)
moveIndices = (Y~=0);
```

```
X = X(moveIndices,:);
Y = Y(moveIndices,:);

whichstats = {'tstat'};
stats = regstats(Y, X, [1 0; 0 1], whichstats);
ts = stats.tstat;
CoeffTable = dataset({ts.beta,'Coef'},{ts.se,'StdErr'}, ...
        {ts.t,'tStat'},{ts.pval,'pVal'})

% Now we trade (by mean reversion)
%-------------------------------------------------
profit = 0;
for i = N:(length(t)-1)
    prevMove = p(i)-p(i-1);
    if(prevMove>0) % Previous move went up,
                   %    predict it will go down,
                   %    so SELL
        profit = profit + (p(i)-p(i+1));
    elseif(prevMove<0) % Previous move went down,
                       %    predict it will go up,
                       %    so BUY
        profit = profit + (p(i+1)-p(i));
    end
end
disp(profit)
```

```
%-------------------------------------------------------------
% Tony Hyun Kim
% CS 246, PS 4, Problem 1(c)
% Hello SVM
%-------------------------------------------------------------
clear all; close all;

n = 1000;
d = 3;
X = randn(n,d);
randnoise = randn(n,1);
Y = ((X(:,1)+X(:,2)+randnoise)>0)*2-1;

% Training set is first 30
n = 30;
Yt = Y(1:n,1);
Xt = X(1:n,:);

C = norm(mean(abs(Xt)));
cvx_begin % Classical SVM
    variables w(d) e(n) b
    dual variable alpha
    minimize( 0.5*w'*w + C*sum(e)) % norm(w) takes an extra sqrt
    subject to
        Yt.*(Xt*w+b)-1+e > 0 : alpha ;
        e>0; % slack
cvx_end
```

```
%-------------------------------------------------------------
% Tony Hyun Kim
% CS 246, PS 4, Problem 1(d)
% SVM model
%-------------------------------------------------------------

% Previously, we computed the trade time and price
%   in the vectors t and p respectively

% Training time is 10 AM to 3 PM, i.e. 5 total hours. Note
%   that t(1) has been set in 'load_file.m' to be 10 AM.
t_tr = t(1) + 5*60*60;
N = find(t_tr<t,1); % Training examples

% As for our data features, we will look at the last K
%   K price differentials
K = 3;

X = zeros(N,K);
Y = zeros(N,1);

shift = zeros(1,K); % Shift register for price differential
for i = 2:N
    dp = p(i)-p(i-1); % Newest observed move
    shift = [dp shift(1:(end-1))]; % Insert into shift reg
    X(i,:) = shift; % Current state of shift register are the features
    Y(i-1) = sign(dp);
end

% Remove sparse entries in X/Y (i.e. no price move recorded)
%-------------------------------------------------------------
moveIndices = (Y~=0);
X = X(moveIndices,:);
Y = Y(moveIndices,:);
N = length(Y);

% Let's try SVM regression
%-------------------------------------------------------------
C = 1e4*norm(mean(abs(X)));
cvx_begin
    variables w(K) e(N) b
    dual variable alpha
    minimize(0.5*w'*w + C*sum(e))
    subject to
        Y.*(X*w+b)-1+e > 0 : alpha;
        e>0;
cvx_end

% Apply resulting model to training set
%-------------------------------------------------------------
Ysvm = X*w+b;
% Ysvm = sign(Ysvm);

% Consider the distribution of our "confidences" in the training set
qu = quantile(abs(Ysvm),[0.75 0.50 0.33]);

% Now we trade (by mean reversion)
%-------------------------------------------------
profit = 0;
remTrades = 10;
shift = zeros(1,K);
for i = N:(length(t)-1)
    if(remTrades==0) % No more trades to perform
        break;
    end

    % Load in new prices
    dp = p(i)-p(i-1);
```

```
    shift = [dp shift(1:(end-1))];

    pred_svm = shift*w+b;
    conf = abs(pred_svm); % Our confidence in the answer (i.e. the margin)

    if(conf>qu(1)) % High confidence.
        units  = min(5,remTrades);
        if(pred_svm>0) % Expect price to go up. Buy.
            profit = profit + units*(p(i+1)-p(i));
        else % Expect price to go down. Sell.
            profit = profit + units*(p(i)-p(i+1));
        end
        remTrades = remTrades - units;
    elseif(conf>qu(2)) % Middle confidence.
        units  = min(2,remTrades);
        if(pred_svm>0) % Expect price to go up. Buy.
            profit = profit + units*(p(i+1)-p(i));
        else % Expect price to go down. Sell.
            profit = profit + units*(p(i)-p(i+1));
        end
        remTrades = remTrades - units;
    elseif(conf>qu(3)) % Low confidence.
        units  = min(1,remTrades);
        if(pred_svm>0) % Expect price to go up. Buy.
            profit = profit + units*(p(i+1)-p(i));
        else % Expect price to go down. Sell.
            profit = profit + units*(p(i)-p(i+1));
        end
        remTrades = remTrades - units;
    end
end
```