MAT 515 Financial Modeling Paper by Oguzcan Adabuk

Part 1

Baum-Welch 2 State Poisson HMM

```
%%% Baum_Welch_Poisson_HMM_v3.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%% Baum-Welch algorithm for Hidden Markov Model
%%%
%%% Estimate parameters for 2-state Poisson-HMM
%%% by using matrices and the Baum-Welch algorithm
%%%
%%% Created on: May 2, 2020
%%% Minor revision on: May 4, 2020
clear all;
%%% Get data for number of major earthquakes in the world
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%% counts of major earthquakes in the world from 1990 to 2006,
%%% a major earthquake is magnitude 7 or above,
%%% source of data is from US Geological Survey
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
row1 = [13 14 8 10 16 26 32 27 18 32 36 24 22 23 22 18 25 21 21 14];
row2 = [8 11 14 23 18 17 19 20 22 19 13 26 13 14 22 24 21 22 26 21];
row3 = [23 24 27 41 31 27 35 26 28 36 39 21 17 22 17 19 15 34 10 15];
row4 = [22 18 15 20 15 22 19 16 30 27 29 23 20 16 21 21 25 16 18 15];
row5 = [18 14 10 15 8 15 6 11 8 7 18 16 13 12 13 20 15 16 12 18];
row6 = [15 16 13 15 16 11 11];
gedata = [row1 row2 row3 row4 row5 row6];
% observed values of X(1), X(2), X(3), X(4), etc.
xvect = gedata;
% initialize parameters for 2-state Poisson-HMM
init_lambda_1 = 10;
init_lambda_2 = 30;
init_dvect = [0.5 0.5]; %% initial distribution for Markov Chain
init_Gmat = [0.9 0.1; 0.1 0.9]; %% transition probability matrix
% matrix to store alpha1_vect, alpha2_vect, etc.
% row 1 contains alpha1_vect
% row 2 contains alpha2_vect
alpha_mat = zeros(107, 2);
% matrix to store beta1_vect, beta2_vect, etc.
beta_mat = zeros(107, 2);
% tensor to store matrices Px1_mat, Px2_mat, etc.
Ptensor = zeros(2,2,108);
vhat = zeros(2, 2, 107);
uhat = zeros(2, 107);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% use initial parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lambda_1 = init_lambda_1;
lambda_2 = init_lambda_2;
dvect = init_dvect;
Gmat = init_Gmat;

iter_arr = int16.empty;
```

```
lambda_1_arr = double.empty;
lambda_2_arr = double.empty;

for iter = 1:31

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % make Px1_mat, Px2_mat, Px3_mat, etc.
    %
    % Px1_mat is a 2 by 2 diagonal matrix for P(x1)
    % Px2_mat is a 2 by 2 diagonal matrix for P(x2)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for t=1:107
        m = xvect(t);

        p11 = (lambda_1)^(m)*exp(-lambda_1)/factorial(m);
        p22 = (lambda_2)^(m)*exp(-lambda_2)/factorial(m);
        Pxt_mat = diag( [p11, p22] );
        Ptensor(:,:,t) = Pxt_mat;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % alpha1_vect = dvect*Px1_mat;
    % alpha2_vect = alpha1_vect*Gmat*Px2_mat;
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    alpha_mat(1,:) = dvect*Ptensor(:,:,1);
    for j = 2:107
        alpha_mat(j,:) = alpha_mat(j-1,:)*Gmat*Ptensor(:,:,j);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % If there are only 7 observations, then these are the backward probabilities:
    %
    % beta7_vect = ones(1,2);
    % beta6_vect = ( Gmat*Px7_mat*ones(2,1) )';
    % beta5_vect = ( Gmat*Px6_mat*beta6_vect' )';
    % beta4_vect = ( Gmat*Px5_mat*beta5_vect' )';
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %str = '';
    beta_mat(107,:) = ones(1,2);
    for t=107:-1:2
        % students should write code here to make backward probabilities
        beta_mat(t-1,:)=Gmat*Ptensor(:,:,t)*beta_mat(t,:)';
    end


    likelihood = alpha_mat(2,:)*beta_mat(2,:)';
    if (abs( likelihood - alpha_mat(3,:)*beta_mat(3,:)' ) > 0.01)
        disp('Warning: likelihood is wrong.');
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % E step
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % update values of vhat
    for j=1:2
        for k=1:2
            for t = 2:107
                expr1 = alpha_mat(t-1,j)*Gmat(j,k)*Ptensor(k,k,t)*beta_mat(t,k);
```

```
                vhat(j,k,t) = expr1/likelihood;
                % remember to include p_k(x_t)
            end
        end
    end
    % update values of uhat
    for j=1:2
        for t=1:107
            uhat(j,t) = alpha_mat(t,j)*beta_mat(t,j)/likelihood;
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % M step
    %
    % maximize CDLL with respect to the three parmaters:
    %  (1) dvect,
    %  (2) entries in Gmat(j,k)
    %  (3) lambda_j for Poisson-Hmm
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % update dvect
    dvect(1) = uhat(1,1);
    dvect(2) = 1- dvect(1); % sum of all delta_j = 1
    % update Gmat(j,k)
    freq = zeros(2,2);
    for j=1:2
        for k=1:2
            freq(j,k) = sum( vhat(j,k,:) );
        end
    end
    for j=1:2
        for k=1:2
            Gmat(j,k) = freq(j,k)/sum(freq(:,k));
        end
    end
    % update lambda_j for Poisson-HMM
    total = 0;
    for t = 1:107
        total = total + uhat(1,t)*xvect(t);
    end
    lambda_1 = total/sum( uhat(1,:) );
    total = 0;
    for t = 1:107
        total = total + uhat(2,t)*xvect(t);
    end
    lambda_2 = total/sum( uhat(2,:) );

    disp(['iter = ', num2str(iter)]);
    disp(['lambda_1 = ', num2str(lambda_1)]);
    disp(['lambda_2 = ', num2str(lambda_2)]);


    if(iter <= 5 || mod(iter,10) == 0)
       iter_arr(length(iter_arr) + 1)=iter;
       lambda_1_arr(length(lambda_1_arr) + 1)=lambda_1;
       lambda_2_arr(length(lambda_2_arr) + 1)=lambda_2;
    end
end

result = '';
s={'|'};
disp(' i | lambda_1 | lambda_2 ');
```

```
for i=1: 8
    result = strcat(num2str(iter_arr(i)), s, num2str(lambda_1_arr(i)),s ,
num2str(lambda_2_arr(i)));
    disp(result);
    result = '';
end

disp('output parameters of Poisson-HMM');
disp(['lambda_1 = ', num2str(lambda_1)]);
disp(['lambda_2 = ', num2str(lambda_2)]);

Gmat
```

Output:

```
i | lambda_1 | lambda_2
    '1|13.7419|24.1691'

    '2|14.0904|24.0611'

    '3|14.2592|24.1545'

    '4|14.4038|24.2875'

    '5|14.5453|24.4502'

    '10|15.1093|25.3575'

    '20|15.3948|25.9618'

    '30|15.4189|26.0141'

output parameters of Poisson-HMM
lambda_1 = 15.4194
lambda_2 = 26.015

Gmat =

    0.9284    0.1190
    0.0716    0.8810
```

Part 2

Baum-Welch 3 State Poisson HMM

```
%%% Baum_Welch_Poisson_HMM_v3.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%
%%% Baum-Welch algorithm for Hidden Markov Model
%%%
%%% Estimate parameters for 2-state Poisson-HMM
%%% by using matrices and the Baum-Welch algorithm
%%%
%%% Created on: May 2, 2020
%%% Minor revision on: May 4, 2020
clear all;
%%% Get data for number of major earthquakes in the world
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%% counts of major earthquakes in the world from 1990 to 2006,
%%% a major earthquake is magnitude 7 or above,
%%% source of data is from US Geological Survey
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
row1 = [13 14 8 10 16 26 32 27 18 32 36 24 22 23 22 18 25 21 21 14];
row2 = [8 11 14 23 18 17 19 20 22 19 13 26 13 14 22 24 21 22 26 21];
row3 = [23 24 27 41 31 27 35 26 28 36 39 21 17 22 17 19 15 34 10 15];
row4 = [22 18 15 20 15 22 19 16 30 27 29 23 20 16 21 21 25 16 18 15];
row5 = [18 14 10 15 8 15 6 11 8 7 18 16 13 12 13 20 15 16 12 18];
row6 = [15 16 13 15 16 11 11];
gedata = [row1 row2 row3 row4 row5 row6];
% observed values of X(1), X(2), X(3), X(4), etc.
xvect = gedata;
% initialize parameters for 3-state Poisson-HMM
init_lambda_1 = 10;
init_lambda_2 = 20;
init_lambda_3 = 30;

init_dvect = [0.3333 0.3333, 0.3334]; %% initial distribution for Markov Chain
init_Gmat = [0.8 0.1 0.1; 0.1 0.8 0.1; 0.1 0.1 0.8]; %% transition probability matrix
% matrix to store alpha1_vect, alpha2_vect, etc.
% row 1 contains alpha1_vect
% row 2 contains alpha2_vect
alpha_mat = zeros(107, 3);
% matrix to store beta1_vect, beta2_vect, etc.
beta_mat = zeros(107, 3);
% tensor to store matrices Px1_mat, Px2_mat, etc.
Ptensor = zeros(3,3,108);
vhat = zeros(3, 3, 107);
uhat = zeros(3, 107);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% use initial parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lambda_1 = init_lambda_1;
lambda_2 = init_lambda_2;
lambda_3 = init_lambda_3;

dvect = init_dvect;
Gmat = init_Gmat;

iter_arr = int16.empty;
lambda_1_arr = double.empty;
lambda_2_arr = double.empty;
lambda_3_arr = double.empty;

for iter = 1:31

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % make Px1_mat, Px2_mat, Px3_mat, etc.
    %
```

```
    % Px1_mat is a 2 by 2 diagonal matrix for P(x1)
    % Px2_mat is a 2 by 2 diagonal matrix for P(x2)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for t=1:107
        m = xvect(t);

        p11 = (lambda_1)^(m)*exp(-lambda_1)/factorial(m);
        p22 = (lambda_2)^(m)*exp(-lambda_2)/factorial(m);
        p33 = (lambda_3)^(m)*exp(-lambda_3)/factorial(m);
        %disp(strcat('p11: ', num2str(p11), ' p22: ', num2str(p22), ' p33: ',
num2str(p33)));

        Pxt_mat = diag( [p11, p22, p33] );
        Ptensor(:,:,t) = Pxt_mat;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % alpha1_vect = dvect*Px1_mat;
    % alpha2_vect = alpha1_vect*Gmat*Px2_mat;
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    alpha_mat(1,:) = dvect*Ptensor(:,:,1);
    for j = 2:107
        alpha_mat(j,:) = alpha_mat(j-1,:)*Gmat*Ptensor(:,:,j);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % If there are only 7 observations, then these are the backward probabilities:
    %
    % beta7_vect = ones(1,2);
    % beta6_vect = ( Gmat*Px7_mat*ones(2,1) )';
    % beta5_vect = ( Gmat*Px6_mat*beta6_vect' )';
    % beta4_vect = ( Gmat*Px5_mat*beta5_vect' )';
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %str = '';
    beta_mat(107,:) = ones(1,3);
    for t=107:-1:2
        % students should write code here to make backward probabilities
        beta_mat(t-1,:)=Gmat*Ptensor(:,:,t)*beta_mat(t,:)';
    end


    likelihood = alpha_mat(3,:)*beta_mat(3,:)';
    if (abs( likelihood - alpha_mat(4,:)*beta_mat(4,:)' ) > 0.01)
        disp('Warning: likelihood is wrong.');
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % E step
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % update values of vhat
    for j=1:3
        for k=1:3
            for t = 2:107
                expr1 = alpha_mat(t-1,j)*Gmat(j,k)*Ptensor(k,k,t)*beta_mat(t,k);
                vhat(j,k,t) = expr1/likelihood;
                % remember to include p_k(x_t)
            end
        end
```

```
end
% update values of uhat
for j=1:3
    for t=1:107
        uhat(j,t) = alpha_mat(t,j)*beta_mat(t,j)/likelihood;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% M step
%
% maximize CDLL with respect to the three parmaters:
% (1) dvect,
% (2) entries in Gmat(j,k)
% (3) lambda_j for Poisson-Hmm
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% update dvect
dvect(1) = uhat(1,1);
dvect(2) = uhat(2,1); % sum of all delta_j = 1
dvect(3) = uhat(3,1);
% update Gmat(j,k)
freq = zeros(3,3);
for j=1:3
    for k=1:3
        freq(j,k) = sum( vhat(j,k,:) );
    end
end
for j=1:3
    for k=1:3
        Gmat(j,k) = freq(j,k)/sum(freq(:,k));
    end
end
% update lambda_j for Poisson-HMM
total = 0;
for t = 1:107
    total = total + uhat(1,t)*xvect(t);
end
lambda_1 = total/sum( uhat(1,:) );

total = 0;
for t = 1:107
    total = total + uhat(2,t)*xvect(t);
end
lambda_2 = total/sum( uhat(2,:) );

total = 0;
for t = 1:107
    total = total + uhat(3,t)*xvect(t);
end
lambda_3 = total/sum( uhat(3,:) );

disp(['iter = ', num2str(iter)]);
disp(['lambda_1 = ', num2str(lambda_1)]);
disp(['lambda_2 = ', num2str(lambda_2)]);
disp(['lambda_3 = ', num2str(lambda_3)]);

if(iter <= 5 || mod(iter,10) == 0)
   iter_arr(length(iter_arr) + 1)=iter;
   lambda_1_arr(length(lambda_1_arr) + 1)=lambda_1;
   lambda_2_arr(length(lambda_2_arr) + 1)=lambda_2;
   lambda_3_arr(length(lambda_3_arr) + 1)=lambda_3;
end
```

```
end

result = '';
s={'|'};
disp('i | lambda_1 | lambda_2 | lambda_3');

for i=1: length(iter_arr)
    result = strcat(num2str(iter_arr(i)), s, num2str(lambda_1_arr(i)),s ,
num2str(lambda_2_arr(i)), s, num2str(lambda_3_arr(i)));
    disp(result);
    result = '';
end

disp('output parameters of Poisson-HMM');
disp(['lambda_1 = ', num2str(lambda_1)]);
disp(['lambda_2 = ', num2str(lambda_2)]);

Gmat
```

Output:

```
i | lambda_1 | lambda_2 | lambda_3
    '1|11.6987|19.0304|29.7408'

    '2|12.2657|19.0901|29.5818'

    '3|12.707|19.3329|29.495'

    '4|12.9661|19.5475|29.5264'

    '5|13.0693|19.6523|29.6082'

    '10|13.1316|19.7158|29.724'

    '20|13.1339|19.7138|29.7099'

    '30|13.1339|19.7137|29.7095'

output parameters of Poisson-HMM
lambda_1 = 13.1339
lambda_2 = 19.7137

Gmat =

    0.9395    0.0214    0.0503
    0.0605    0.9064    0.1399
    0.0000    0.0722    0.8098
```

Part 3

(a) Baum-Welch 3 State Normal HMM

Initially my code was producing NaN values. I figured this happened because of an undeflow in *beta_mat*. *Beta_mat* was getting filled with 0s after 400 something iterations. As a result this caused the *likelihood* to be 0 in line 108. Since *uhat* and *vhat* are calculated with *likelihood* as the denominator, a chain production of NaN values took over the program like Agent Smith took over the Matrix. I know the clean way to handle this to apply log function but in limited time I had left, I took the easy way of dealing with this problem and I dropped the first 25 rows of data(25 out of 450, I can live with that).

```
clear all;

fileID = fopen('ftse100.txt');
ftse100 = fscanf(fileID, '%f');

xvect = ftse100;
xvect = xvect(25:end); %Drop the first 25 rows because Underflow!

n = length(xvect);

disp(mean(xvect));
disp(std(xvect));
% initialize parameters for 3-state Normal-HMM
init_mu_1 = -0.2;
init_mu_2 = 0.025;
init_mu_3 = 0.2;

init_sigma_1 = 1.363257;
init_sigma_2 = 1.363257;
init_sigma_3 = 1.363257;

init_dvect = [0.3333 0.3333, 0.3334]; %% initial distribution for Markov Chain
init_Gmat = [0.8 0.1 0.1; 0.1 0.8 0.1; 0.1 0.1 0.8]; %% transition probability matrix
% matrix to store alpha1_vect, alpha2_vect, etc.
% row 1 contains alpha1_vect
% row 2 contains alpha2_vect
alpha_mat = zeros(n, 3);
% matrix to store beta1_vect, beta2_vect, etc.
beta_mat = zeros(n, 3);
% tensor to store matrices Px1_mat, Px2_mat, etc.
Ptensor = zeros(3,3,n+1);
vhat = zeros(3, 3, n);
uhat = zeros(3, n);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% use initial parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mu_1 = init_mu_1;
mu_2 = init_mu_2;
mu_3 = init_mu_3;

sigma_1 = init_sigma_1;
sigma_2 = init_sigma_2;
sigma_3 = init_sigma_3;

dvect = init_dvect;
Gmat = init_Gmat;
```

```
iter_arr = int16.empty;
mu_1_arr = double.empty;
mu_2_arr = double.empty;
mu_3_arr = double.empty;

sigma_1_arr = double.empty;
sigma_2_arr = double.empty;
sigma_3_arr = double.empty;

for iter = 1:16

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % make Px1_mat, Px2_mat, Px3_mat, etc.
    %
    % Px1_mat is a 2 by 2 diagonal matrix for P(x1)
    % Px2_mat is a 2 by 2 diagonal matrix for P(x2)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for t=1:n
        m = xvect(t);

        p11 = normal_pdf(m,mu_1, sigma_1);
        p22 = normal_pdf(m,mu_2,sigma_2);
        p33 = normal_pdf(m,mu_3,sigma_3);

        %disp(strcat('p11: ', num2str(p11), ' p22: ', num2str(p22), ' p33: ',
num2str(p33)));
        Pxt_mat = diag( [p11, p22, p33] );
        Ptensor(:,:,t) = Pxt_mat;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % alpha1_vect = dvect*Px1_mat;
    % alpha2_vect = alpha1_vect*Gmat*Px2_mat;
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    alpha_mat(1,:) = dvect*Ptensor(:,:,1);
    for j = 2:n
        alpha_mat(j,:) = alpha_mat(j-1,:)*Gmat*Ptensor(:,:,j);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %
    % If there are only 7 observations, then these are the backward probabilities:
    %
    % beta7_vect = ones(1,2);
    % beta6_vect = ( Gmat*Px7_mat*ones(2,1) )';
    % beta5_vect = ( Gmat*Px6_mat*beta6_vect' )';
    % beta4_vect = ( Gmat*Px5_mat*beta5_vect' )';
    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %str = '';
    beta_mat(n,:) = ones(1,3);
    for t=n:-1:2
        % students should write code here to make backward probabilities
        beta_mat(t-1,:) = Gmat * Ptensor(:,:,t) * beta_mat(t,:)';
    end


    likelihood = alpha_mat(3,:)*beta_mat(3,:)';
```

```
if (abs( likelihood - alpha_mat(4,:)*beta_mat(4,:)' ) > 0.01)
    disp('Warning: likelihood is wrong.');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% E step
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% update values of vhat
for j=1:3
    for k=1:3
        for t = 2:n
            expr1 = alpha_mat(t-1,j)*Gmat(j,k)*Ptensor(k,k,t)*beta_mat(t,k);
            vhat(j,k,t) = expr1/likelihood; %likelihood -> LT
            % remember to include p_k(x_t)
        end
    end
end
% update values of uhat
for j=1:3
    for t=1:n
        uhat(j,t) = alpha_mat(t,j)*beta_mat(t,j)/likelihood;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% M step
%
% maximize CDLL with respect to the three parmaters:
% (1) dvect,
% (2) entries in Gmat(j,k)
% (3) lambda_j for Poisson-Hmm
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% update dvect
dvect(1) = uhat(1,1);
dvect(2) = uhat(2,1); % sum of all delta_j = 1
dvect(3) = uhat(3,1);
% update Gmat(j,k)
freq = zeros(3,3);
for j=1:3
    for k=1:3
        freq(j,k) = sum( vhat(j,k,:) );
    end
end
for j=1:3
    for k=1:3
        Gmat(j,k) = freq(j,k)/sum(freq(:,k));
    end
end

% update mu_j for Normal-HMM
tot_xt = 0;
for t = 1:n
    tot_xt = tot_xt + uhat(1,t)*xvect(t);
end
mu_1 = tot_xt/sum( uhat(1,:) );

tot_xt = 0;
for t = 1:n
    tot_xt = tot_xt + uhat(2,t)*xvect(t);
end
mu_2 = tot_xt/sum( uhat(2,:) );
```

```
    tot_xt = 0;
    for t = 1:n
        tot_xt = tot_xt + uhat(3,t)*xvect(t);
    end
    mu_3 = tot_xt/sum( uhat(3,:) );

    % update sigma_j for Normal-HMM
    tot = 0;
    for t = 1:n
        tot = tot + uhat(1,t)*(xvect(t)-mu_1)^2;
    end
    sigma_1 = sqrt(tot/sum( uhat(1,:) ));

    tot = 0;
    for t = 1:n
        tot = tot + uhat(2,t)*(xvect(t)-mu_2)^2;
    end
    sigma_2 = sqrt(tot/sum( uhat(2,:) ));

    tot = 0;
    for t = 1:n
        tot = tot + uhat(3,t)*(xvect(t)-mu_3)^2;
    end
    sigma_3 = sqrt(tot/sum( uhat(3,:) ));

    disp(['iter = ', num2str(iter)]);
    disp(['mu_1 = ', num2str(mu_1)]);
    disp(['mu_2 = ', num2str(mu_2)]);
    disp(['mu_3 = ', num2str(mu_3)]);

    if(iter < 5 || mod(iter,5) == 0)
       iter_arr(length(iter_arr) + 1)=iter;
       mu_1_arr(length(mu_1_arr) + 1)= mu_1;
       mu_2_arr(length(mu_2_arr) + 1)= mu_2;
       mu_3_arr(length(mu_3_arr) + 1)= mu_3;

       sigma_1_arr(length(sigma_1_arr) + 1)= sigma_1;
       sigma_2_arr(length(sigma_2_arr) + 1)= sigma_2;
       sigma_3_arr(length(sigma_3_arr) + 1)= sigma_3;
    end
end

result = '';
s={'|'};
disp('i | mu_1 | mu_2 | mu_3 | sigma_1 | sigma_2 | sigma_3');

for i=1: length(iter_arr)
    result = strcat(num2str(iter_arr(i)), s, num2str(mu_1_arr(i)),s ,
num2str(mu_2_arr(i)), s, num2str(mu_3_arr(i)), s, num2str(sigma_1_arr(i)), s,
num2str(sigma_2_arr(i)), s, num2str(sigma_3_arr(i)));
    disp(result);
    result = '';
end

disp('output parameters of Poisson-HMM');
disp(['mu_1 = ', num2str(mu_1)]);
disp(['mu_2 = ', num2str(mu_2)]);
disp(['mu_3 = ', num2str(mu_3)]);
disp(['sigma_1 = ', num2str(sigma_1)]);
disp(['sigma_2 = ', num2str(sigma_2)]);
disp(['sigma_3 = ', num2str(sigma_3)]);
Gmat
```

normal_pdf.m

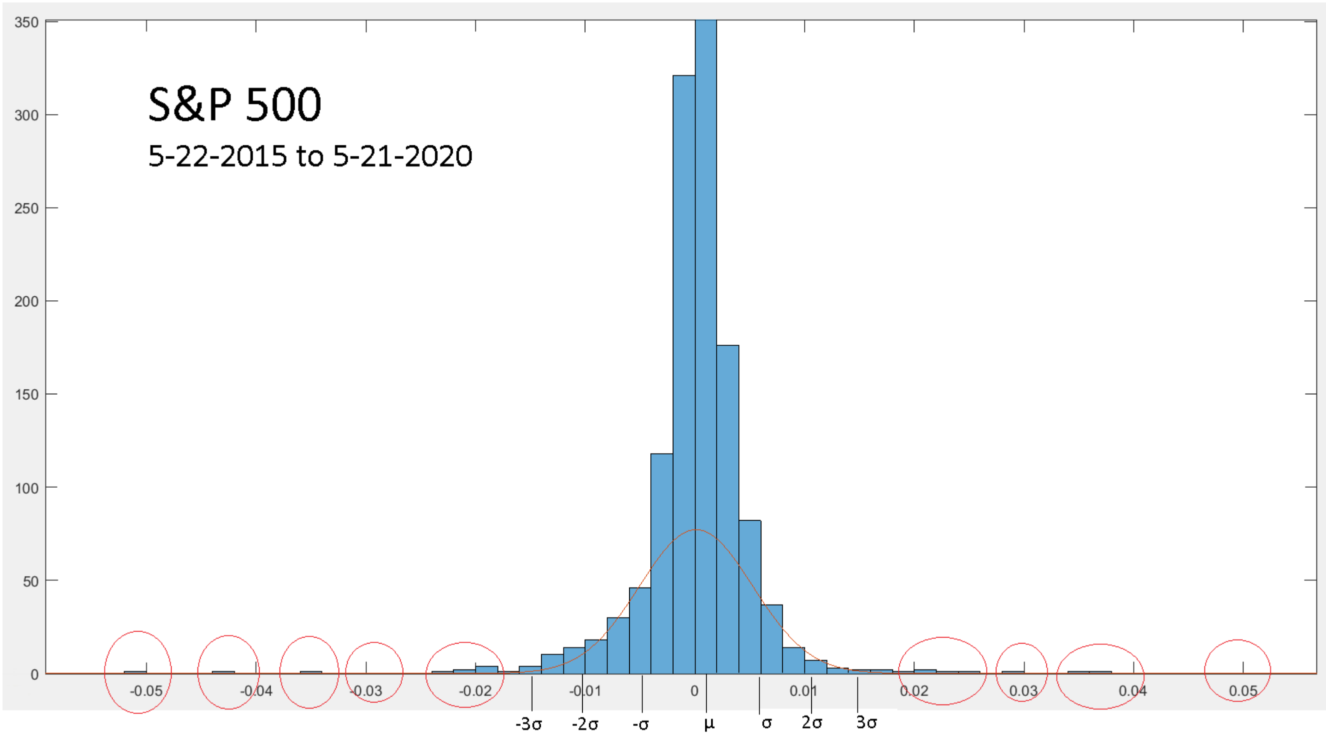
```
function np = normal_pdf(x, mu, sigma)
f = @(m,s,x) 1/sqrt(2*pi*s^2) * exp(-(x-m).^2 / (2*s^2));
np = feval(f, mu, sigma, x);
```

Output:

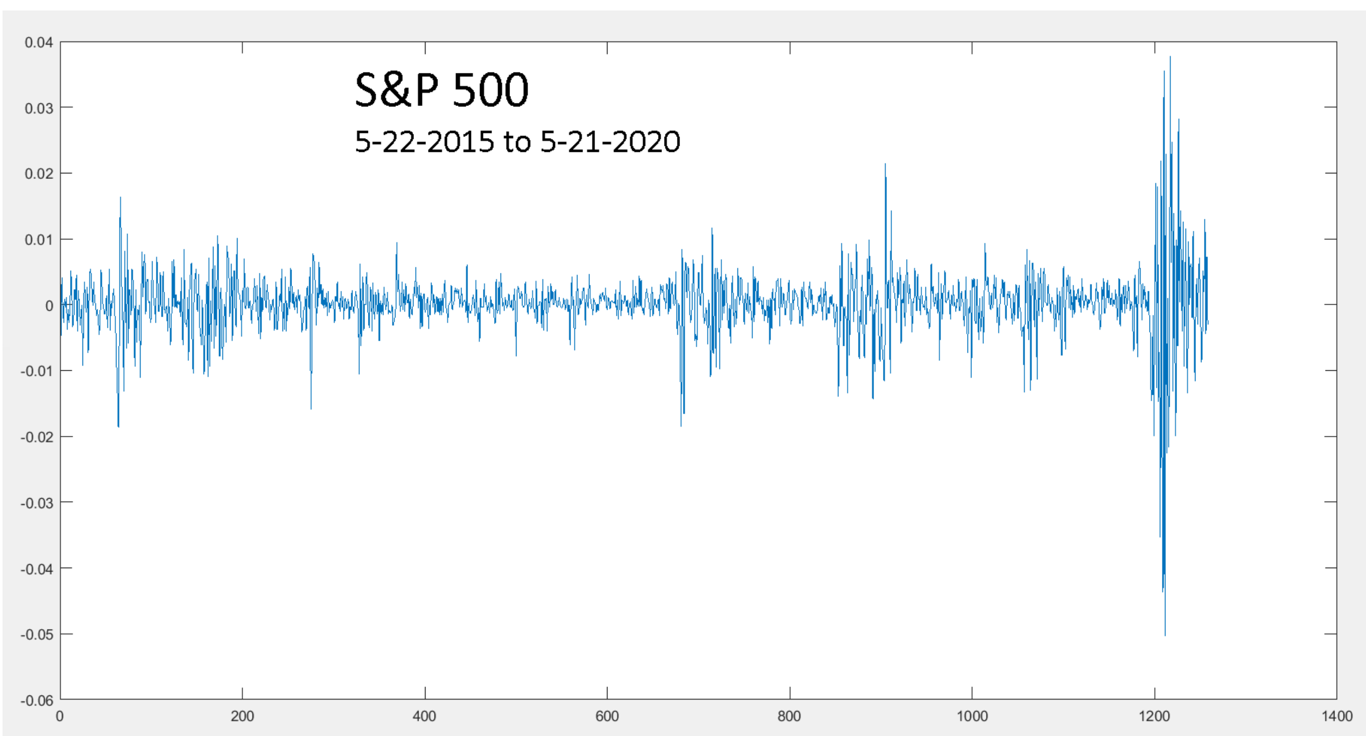
```
   i | mu_1 | mu_2 | mu_3 | sigma_1 | sigma_2 | sigma_3
   '1|-0.23828|0.083807|0.19654|1.8218|1.1296|1.0464'

   '2|-0.60399|0.088299|0.1868|2.7149|0.87916|0.8155'

   '3|-1.0957|0.051016|0.19529|3.4904|0.89608|0.80394'

   '4|-1.3925|-0.006185|0.21529|4.0608|0.93377|0.79598'

   '5|-1.5048|-0.059942|0.24925|4.2844|0.95764|0.77232'

   '10|-1.5254|-0.21905|0.33762|4.3769|0.9823|0.71585'

   '15|-1.5207|-0.26194|0.35078|4.3814|0.97719|0.71345'

output parameters of Poisson-HMM
mu_1 = -1.5198
mu_2 = -0.2673
mu_3 = 0.3525
sigma_1 = 4.3813
sigma_2 = 0.97649
sigma_3 = 0.71289

Gmat =

   0.9179   0.0094   0.0015
   0.0821   0.8533   0.0919
   0.0000   0.1373   0.9065
```

(b) Why normal distribution is not good for modeling stock price returns?

At first glance, the stock market returns seem like normally distributed but they are not.
Main reason is that the outliers that are caused by extreme events happen more often than the normal distribution accounts for. This causes return distribution curve to have "fat tails".
Plain normal distribution models trivialize the risk of extreme events.
In the graph below I used daily returns for the **S&P 500** index for the last 5 years.
The data is from 5-22-2015 to 5-21-2020. I overlayed normal distribution curve over the returns histogram.
Many outliers can be seen far away from the mean that are left out by the normal distribution curve.
It is also obvious that substantial amount of returns are way above the curve.



Second problem is that normal distribution cannot take trends into consideration.
In the second figure you can see different trends in separate time intervals.
It is not a good idea to discard these trends and cramp all data from very different times under one curve.
In the graph below you can see the COVID19 crash that happened in March 2020 (Around 1200).
A model such as HMM that takes time series and trends into consideration is much superior compared to normal distribution for modeling stock returns. It mathematically doesn't make sense to give the same weight to data from a year ago as the data from a week ago.



S&P 500 Analysis Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Analyzing 5 year SPY 500 daily returns between 5-22/2015 and 5-21-2020
% Oguzcan Adabuk
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;

fileID = fopen('spy500.txt');
ftse100 = fscanf(fileID, '%f');

xvect = ftse100;
n = length(xvect);

histogram(xvect);
%plot(xvect);
hold on

mu = mean(xvect);
sigma = std(xvect);

disp(mu);
disp(sigma);

sds=[-3*sigma+mu, -2*sigma+mu, -1*sigma+mu, 0, sigma+mu, 2*sigma+mu, 3*sigma+mu];
disp('standard deviations');
disp(sds);

iter = -0.15+mu;
x=double.empty;
x(1)=iter;
while iter < 0.15
    iter = iter + 0.00001;
    x(length(x) + 1)= iter;
end
y = exp(- 0.5 * ((x - mu) / sigma) .^ 2) / (sigma * sqrt(2 * pi));
plot(x, y);
```

S&P 500 index 5 year daily return data is attached.

(c)

Unfortunately I ran out of time for this section, I probably should have spent less time playing with S&P 500 data and completed this part. However, this paper was very interesting and a lot of fun, you can be sure that I will apply this model to all kinds of financial data with different configurations, thank you!