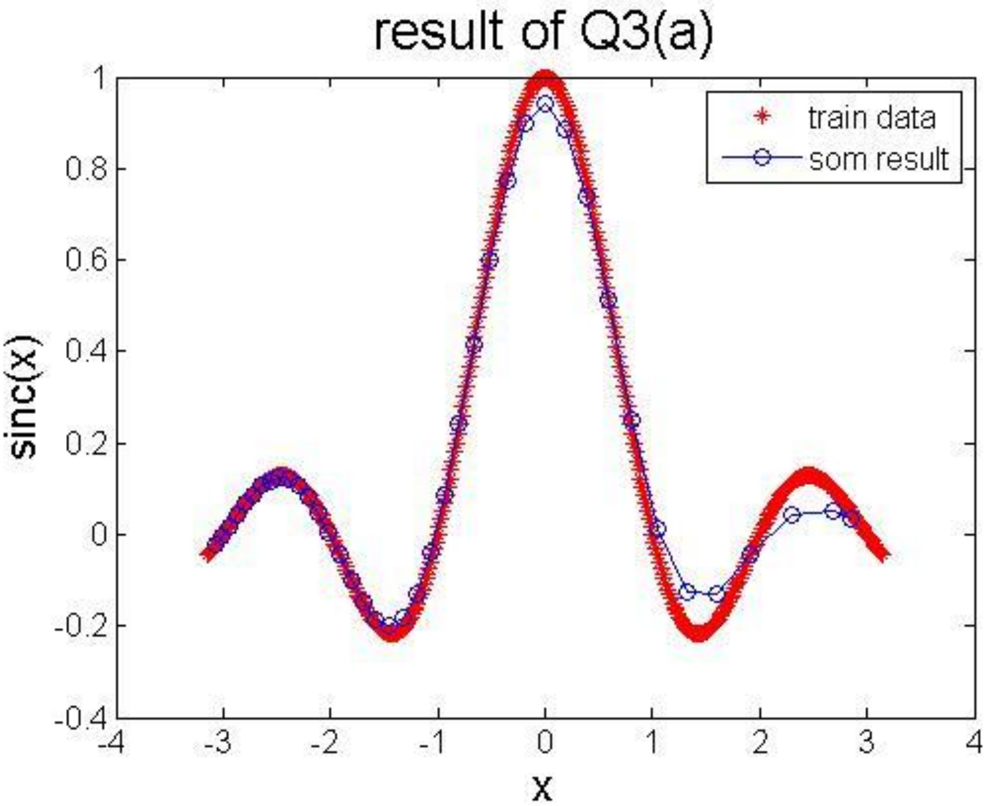
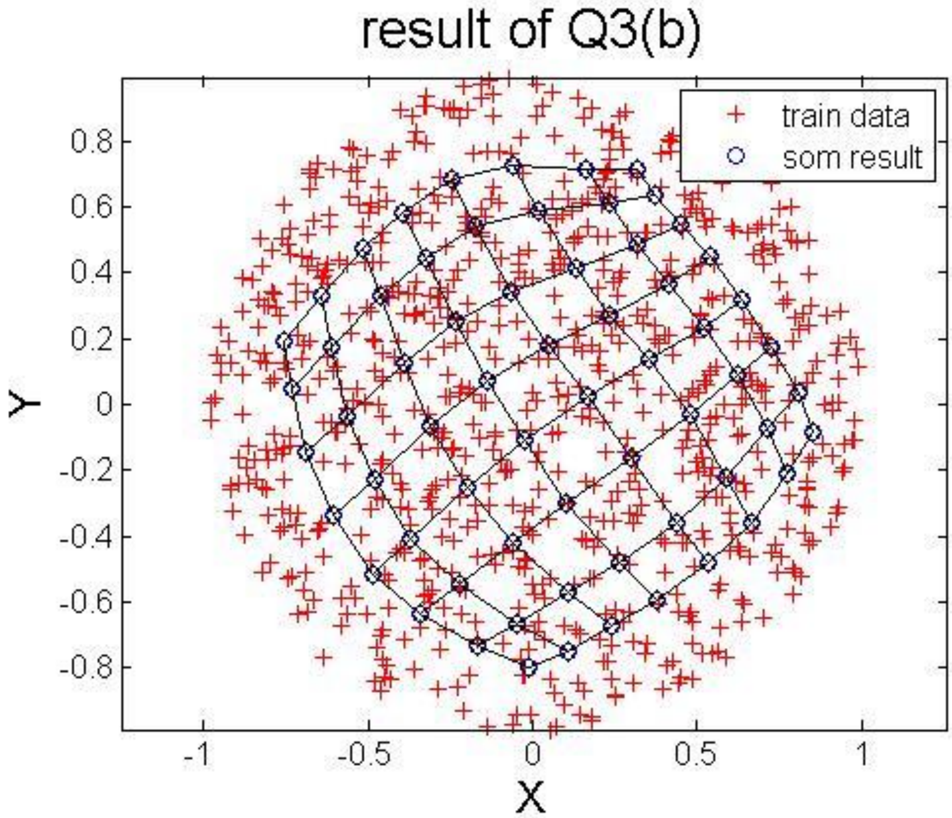
Q2.

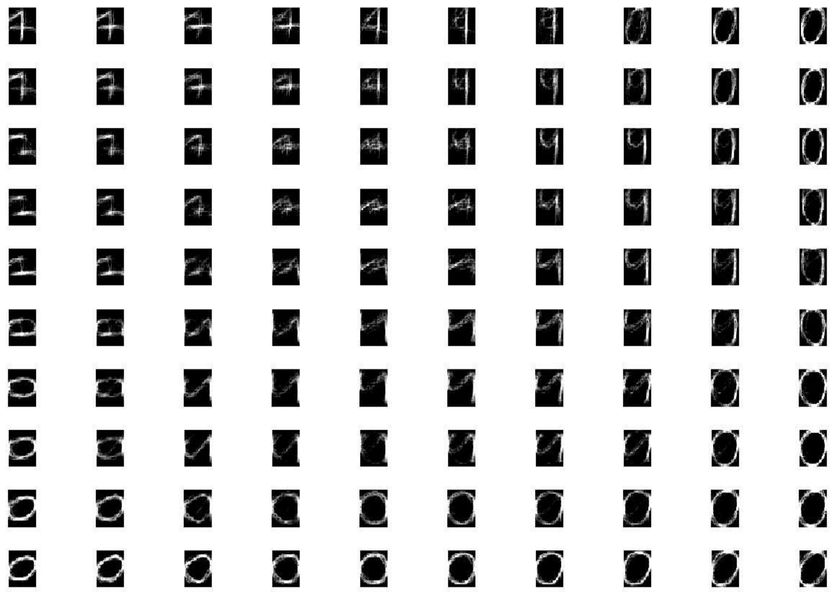
a) Display the trained weights of each output neuron as points in a 2D plane, and plot lines to connect every topological adjacent neurons.
The result is shown below.



b) Display the trained weights of each output neuron as a point in the 2D plane, and plot lines to connect every topological adjacent neurons
The result is shown below.



c) My matric number is A0191818W, ignore classes mod(1,5)=1 and mod(8,5)=3
C-1)Print out corresponding conceptual/semantic map of the trained SOM (as described in page 24 of lecture six) and visualize the trained weights of each output neuron on a 10×10 map.
The result is shown below. The picture could be divide into three parts.



C-2) Apply the trained SOM to classify the test images. The classification can be done in the following fashion: input a test image to SOM, and find out the winner neuron; then label the test image with the winner neuron's label. Calculate the classification accuracy on the whole test set and discuss your findings.
The classification accuracy = 76.67%
The classification is quite well, but also makes some mistakes. Maybe more neurons could improve the accuracy.

The following is the MATLAB code of Q3_1:

```
%%
%Input
clc
clear all;
close all;
x = linspace(-pi,pi,400);
trainX = [x; sinc(x)]; % 2x400 matrix
Index = randperm(size(trainX,1));
trainX_random = trainX(Index,:);
% plot(trainX(1,:),trainX(2,:),'+r');% axis equal

iteration = 500;%iteration
learning_rate_0 = 0.1;%initial learning rate
learning_rate = learning_rate_0;
effective_width_0 = 1;
effective_width = effective_width_0;
time_constant = iteration/log(effective_width_0);

no_neuron = 40;
w = rand(40,2);%randomly initialise all weights
[I,J] = ind2sub([1,no_neuron],1:40);%the positions of neurons in the
som

%%
%Caculation
for n=1:iteration
    for i=1:400
        [~,winIdx] = min(dist(trainX_random(:,i)',w'));
        [winrow,wincolumn] = ind2sub([1,40],winIdx);
        win = [winrow,wincolumn];
        d = exp(-sum(([I(:) J(:)] -
repmat(win,40,1)).^2,2)/(2*effective_width^2));
        for j=1:no_neuron
            w(j,:) = w(j,:) + learning_rate*d(j).*(trainX_random(:,i)'
- w(j,:));
        end
    end
    learning_rate = learning_rate_0*exp(-n/iteration);
    effective_width = effective_width_0*exp(-n/time_constant);
end
%%
%Output
```

```
figure
plot(trainX(1,:),trainX(2,:),'*r',w(:,1),w(:,2),'o-bl');
legend('train data','som result');
set(gca,'FontSize',12);
xlabel('x','FontSize',16);
ylabel('sinc(x)','FontSize',16);
title('result of Q3(a)','FontSize',20);
```

The following is the MATLAB code of Q3_2:

```
%%
%Input
clc
clear all;
close all;
X = randn(800,2);
s2 = sum(X.^2,2);
trainX = (X.*repmat(1*(gammainc(s2/2,1).^(1/2))./sqrt(s2),1,2))';

iteration = 500;%iteration
learning_rate_0 = 0.1;%initial learning rate
learning_rate = learning_rate_0;
effective_width_0 = 1;
effective_width = effective_width_0;
time_constant = iteration/log(effective_width_0);

no_neuron = 64;
w = rand(64,2);%randomly initialise all weights
[I,J] = ind2sub([8,8],1:64);%the positions of neurons in the som

%%
%Caculation
for n=1:iteration
    for i=1:400
        [~,winIdx] = min(dist(trainX(:,i)',w'));
        [winrow,wincolumn] = ind2sub([8,8],winIdx);
        win = [winrow,wincolumn];
        d = exp(-sum(([I(:) J(:)] -
repmat(win,64,1)).^2,2)/(2*effective_width^2));
        for j=1:no_neuron
            w(j,:) = w(j,:) + learning_rate*d(j).*(trainX(:,i)' -
w(j,:));
        end
    end
    learning_rate = learning_rate_0*exp(-n/iteration);
    effective_width = effective_width_0*exp(-n/time_constant);
end
%%
%Output
figure
plot(trainX(1,:),trainX(2,:),'+r',w(:,1),w(:,2),'obl');
axis equal;
```

```
hold on;
for i = 0:7
   plot(w(i*8+1:(i+1)*8,1),w(i*8+1:(i+1)*8,2),'-dk');
end
hold on;
for i = 1:8
   plot(w(i:8:i+56,1),w(i:8:i+56,2),'-dk');
end
hold off;
legend('train data','som result');
set(gca,'FontSize',12);
xlabel('X','FontSize',16);
ylabel('Y','FontSize',16);
title('result of Q3(b)','FontSize',20);
```

The following is the MATLAB code of Q3_3_a:

```
%%
%Input
clc
clear all;
close all;
load('Digits.mat');
%omit the specific class 1 3
Index_train = find(train_classlabel==1|train_classlabel==3);
Index_test = find(test_classlabel==1|test_classlabel==3);
train_data(:,(Index_train)) = [];
train_classlabel(:,(Index_train)) = [];
test_data(:,(Index_test)) = [];
test_classlabel(:,(Index_test)) = [];
%transform to double
train_data = double(train_data);
train_classlabel = double(train_classlabel);
test_data = double(test_data);
test_classlabel = double(test_classlabel);

iteration = 1000;%iteration
learning_rate_0 = 0.1;%initial learning rate
learning_rate = learning_rate_0;
effective_width_0 = 1;
effective_width = effective_width_0;
time_constant = iteration/log(effective_width_0);
w = rand(100,784);%randomly initialise all weights
[I,J] = ind2sub([10,10],1:100);%the positions of neurons in the som
no_neuron = 100;

%%
%Caculation
for n=1:iteration
    for i=1:600
       [~,winIdx] = min(dist(train_data(:,i)',w'));
       [winrow,wincolumn] = ind2sub([10,10],winIdx);
       win = [winrow,wincolumn];
       d = exp(-sum(([I(:) J(:)] -
repmat(win,100,1)).^2,2)/(2*effective_width^2));
       for j=1:no_neuron
          w(j,:) = w(j,:) + learning_rate*d(j).*(train_data(:,i)' -
w(j,:));
       end
```

```
    end
    learning_rate = learning_rate_0*exp(-n/iteration);
    effective_width = effective_width_0*exp(-n/time_constant);
end
%%
%Output
for i = 1:100
        subplot(10,10,i);
        imshow(reshape(w(i,:),[28 28]));
end
save('result');
```

The following is the MATLAB code of Q3_3_b:

```
%Input
clc
clear all;
close all;
load('result.mat');
%caculate the winner weight label
for k = 1:100
    for r = 1:600
        distance(r) = (w(k,:)-train_data(:,r)')*(w(k,:)-
train_data(:,r)')';
    end
    point = find(distance==min(distance));
    win_index(k) = train_classlabel(point(1));
end

for k = 1:60
    for r = 1:100
        distance_test(r) = (w(r,:) - test_data(:,k)')*(w(r,:) -
test_data(:,k)')';
    end
    point = find(distance_test==min(distance_test));
    test_index(k) = win_index(point(1));
end

%%
%Output
number = 0;
for i =1:60
    if test_index(i) == test_classlabel(i)
        number = number+1;
    end
end
accuracy = number/60;
```