# MATH 285 HW2

Xiaoyan Chong

November 23, 2015

# 1 Problem 1

(a) Figure 1 shows the map of Chinese Cities using build-in function *cmdscale.m*.

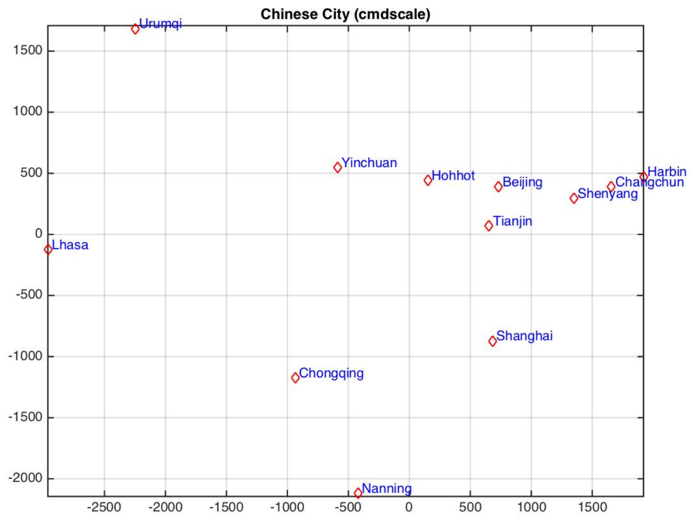


Figure 1: Chinese Cities (cmdscale)

Figure 2 shows the map of Chinese Cities using build-in function *xcmds.m*. By this funciton, I also get $Stress = 0.0805$. Since the stress is less than 0.1, we conclude the result is pretty good.

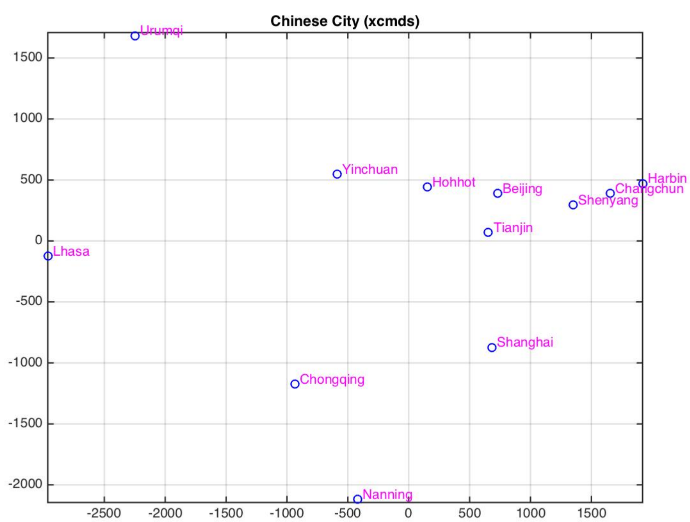


Figure 2: Chinese Cities (xcmds)

The following is the Matlab code:

```
%% load data
load ChineseCityData.mat

%% plot with cmdscale
[Y, e] = cmdscale(dists)

f1_1 = figure;
labels = Cities;
plot(-Y(:,1),-Y(:,2),'rd');
axis equal;
text(-(Y(:,1))+30,-(Y(:,2))+30,labels,'Color','b','HorizontalAlignment','left');
title('Chinese City (cmdscale)');
grid on;

saveas(f1_1,'/Users/XC/Dropbox/SJSU/Courses/Math 285 Data Modeling/HW/hw2/1_cmdscale.jpg','jpg')

%% plot with my own function

[Y, stress]  = xcmds(dists,2)
f1_2 = figure;
labels = Cities;
plot(Y(:,1),Y(:,2),'bo')
axis equal;
text(Y(:,1)+40,Y(:,2)+30,labels,'Color','m','HorizontalAlignment','left');
title('Chinese City (xcmds)');
grid on;

saveas(f1_2,'/Users/XC/Dropbox/SJSU/Courses/Math 285 Data Modeling/HW/hw2/1_xcmds.jpg','jpg')

```

Figure 3: Problem 1 Code

```
function [Y,stress] = xcmds(X,k)
% this is a function for mds
% we can compare this with cmdscale in Matlab
[D,N] = size(X);
Xsquare = X.* X ;
unit1 = ones(D,1);
v = ones(D,1);
I = diag(v);
J = I- 1/D * unit1 * unit1';
B = (-1/2)*J * Xsquare * J;

% extract engivalues and eigenvectors of B
[V,D] = eig(B);
D2 = diag(sort(diag(D),'descend'));
e=diag(D2);
[c,ind] = sort(diag(D),'descend');
V2 = V(:,ind)
eigvaluek = D2(1:k,1:k);eigvectork = V2(:, 1:k);
eigvaluek

% Build Y
Y = eigvectork * sqrt(eigvaluek);

% stress
d = L2_distance(Y',Y')
stress = sqrt(sum(sum((X-d).^2))/sum(sum(X.^2)))
end
```

Figure 4: xcmds.m

(b) No, we could not use these distances to construct a world map. In this case, when we project the distance from spherical onto 2 dimensional space, we could not preserve the distance between two cities. If on spherical, the distance between two cities are pretty close, when projecting to 2-D plane, we cannot preserve the closeness, they may be far awary from each other.

## 2 Problem 2

(1) Data set:Glassdata

(2) These data belong to a Glass Identification Database and were downloaded from the UCI Machine Learning Repository [Newman, et al., 1998].

(3) This data has 214 observations, with 9 variables: refractive index, sodium, magnesium, aluminum, silicon, potassium, calcium, barium, and iron.

(4) The data set also includes a class label: building windows float processed(type 1), building windows not float processed(type 2), vehicle windows float processed(type 3), building windows not float processed (none in this dataset)(type 4), containers(type 5), tableware(type 6), and headlamps(type 7).

(5) One could use this data set to develop a classifier that could be used in forensic science applications.

(6) By the description on (4), we may see type 1,2,3 are very similar, and type 6, 7 are very similar. According to Figure 7 (Isomap result in 2-D), we could see that 1,2 , 3 are grouped together, and 6, 7 are in a different side of (type 1,2, and 3). Since different types are classified based on the concentrations of variables in introduced in (3), we could say there is a pattern, because similar types of glass contains similar concentrations of Na, Mg, Al, Si, K, Ca, Ba, and Fe. And in our case, these similar types are very close to each other in Isomap result.

(7) Although this data set is not as good as figure data set, we could still see there is a pattern. But if we use a figure dataset, it is much easier to detect a pattern.

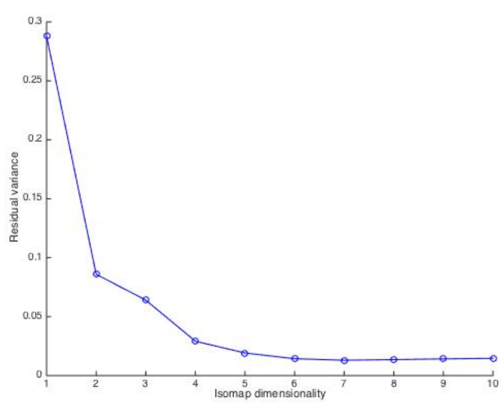


Figure 5: Scree Plot

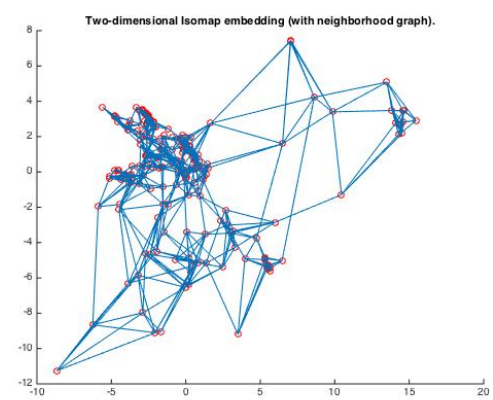


Figure 6: 2-D Isomap embedding

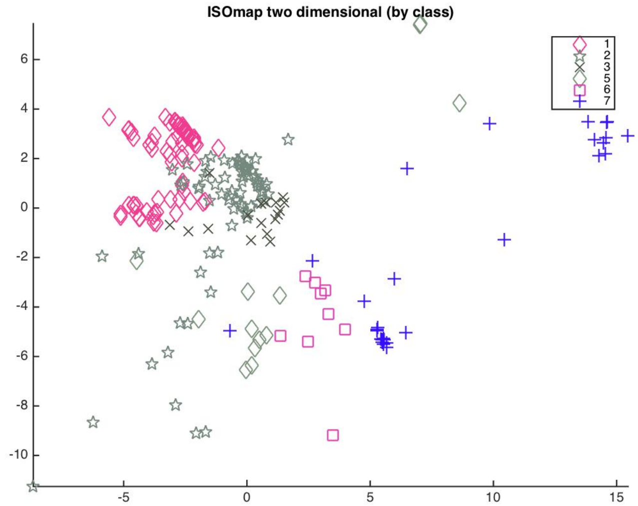


Figure 7: ISOmap

The code is as follows.

Problem_2.m

```
%% Problem 2
clear
close all
clc
%% load glass dataset
glassdata = importdata('glass.txt',',', 0);

%%
% Then get the interpoint dissimilarity matrix.
% We will use standardized Euclidean distance.
tmp = pdist(glassdata,'seuclidean');

% Now put it into a square matrix.
D = squareform(tmp);

% Now we do ISOMAP.
% We will define the neighborhood using the number of nearest neighbors, k = 5.
[Y,R,E] = Isomap(D,'k',5);

lables = zeros(size(glassdata,1),1);
lables(glassdata(:,11)==1) = '1';
lables(glassdata(:,11)==2) = '2';
lables(glassdata(:,11)==3) = '3';
lables(glassdata(:,11)==5) = '5';
lables(glassdata(:,11)==6) = '6';
lables(glassdata(:,11)==7) = '7';

f2_1 = figure; gcplot(Y.coords{2}',lables); axis equal
legend('1','2','3','5','6','7')
title 'ISOmap two dimensional (by class)'
saveas(f2_1,'/Users/XC/Dropbox/SJSU/Courses/Math 285 Data Modeling/HW/hw2/2_isomap.jpg','jpg')
```

Figure 8: Code part 1

```
%% compare with mds

Y_mds = mds(D, 3);
lables = zeros(size(glassdata,1),1);
lables(glassdata(:,11)==1) = '1';
lables(glassdata(:,11)==2) = '2';
lables(glassdata(:,11)==3) = '3';
lables(glassdata(:,11)==5) = '5';
lables(glassdata(:,11)==6) = '6';
lables(glassdata(:,11)==7) = '7';
f2_2 = figure; gcplot(Y_mds,lables);
legend('1','2','3','5','6','7')
title 'mds two dimensional (by class)'

saveas(f2_2,'/Users/XC/Dropbox/SJSU/Courses/Math 285 Data Modeling/HW/hw2/2_mds.jpg','jpg')
```

Figure 9: Code part 2

# 3 Problem 2: Digit Dataset

(1) Data: I use the dataset from homework one, the digit datset (mnist-digit1.mat). I picked the first 1000 data points, and did the following analysis based on these 1000 points. Figure 10 is the scatter plot of original data.

(2) According to Figure 11, we conclude there is a pattern in this figure. For digits on the upper left, they are thinner, and pretty much vertical (only a few have a direction: positive slope). For the digits on the upper right, they are also thinner, but they have a direction (negative slope). For digits on the bottom left, they are thicker, and they are vertical(only several have a direction: positive slope). For digits on the bottom right, they are thicker, and they also have a direction(negative slope).

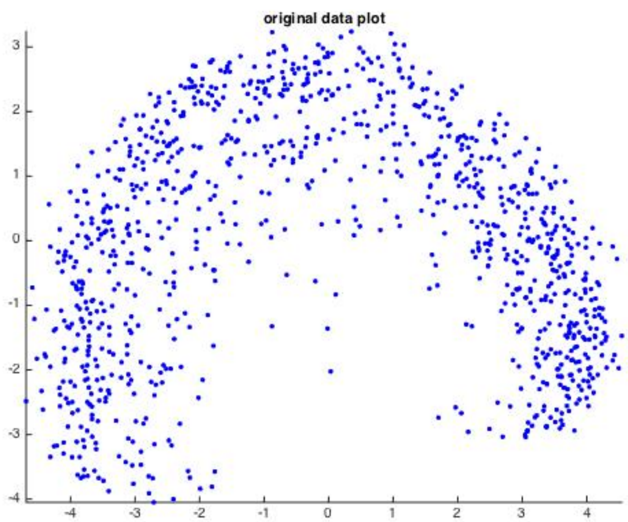


Figure 10: Original Data Scatter

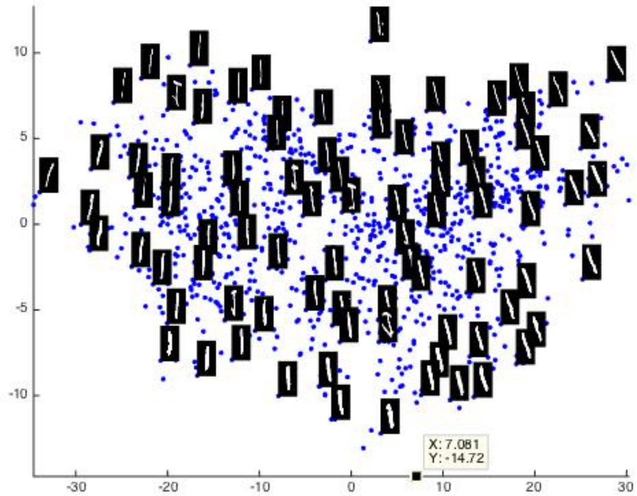


Figure 11: ISOmap

(3) Code, as shown in Figure 12

```
Problem_2.m

%% Problem 2 Digit Dataset
clear
close all
clc

%% load data
load mnist_digit1.mat

%% pick 1000 observations
X_test = X(1:1000,:);

% original data plot
figure; gcplot(X_test)
title 'original data plot'

%% isomap
D = L2_distance(X_test',X_test',1);
[Y,R,E] = Isomap(D,'k',5);
figure; gcplot(Y.coords{2}');

%% add image to the figure
cursor_info; a =cursor_info.Position;hold on; ...
imagesc([a(1) a(1)+2], [a(2) a(2)+2], reshape(X_test(cursor_info.DataIndex,:), 28,28)); ...
figure(gcf); colormap gray
```

Figure 12: Code part

## 4 Problem 3

(1) O $\rightarrow$ A: 2

(2) O $\rightarrow$ B: 4

(3) O$\rightarrow$ C: 4

(4) O $\rightarrow$ D: 8

(5) O$\rightarrow$ E: 7

(6) O $\rightarrow$ F: 14

(7) O $\rightarrow$ T: 13

## 5 Problem 4

Figure 13 is the original data scatter plot.

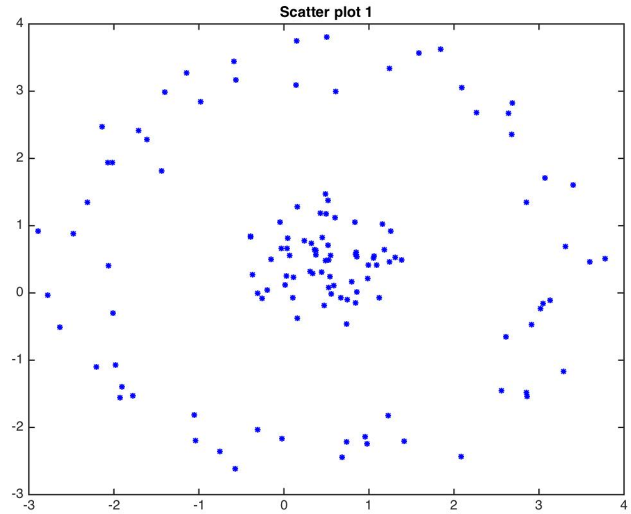


Figure 13: Scatter plot of original

Figure 14 is the plot for Gaussian Kernel PCA 2-D.

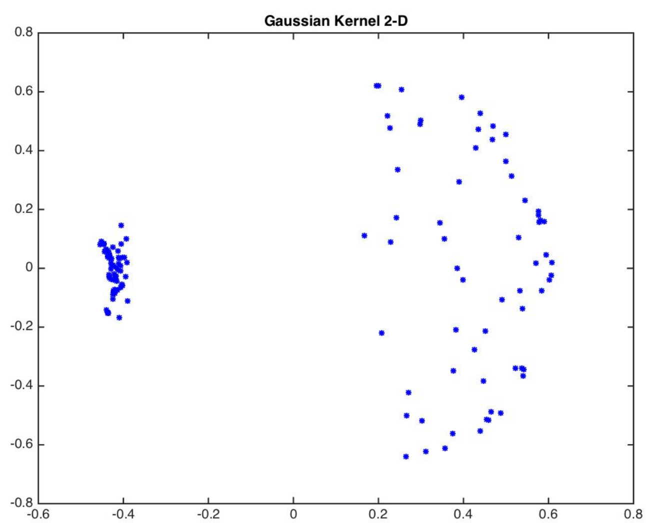


Figure 14: Gaussian Kernel

Comparing the above two figures, we could find that by using Gaussian Kernel PCA, the two groups could be seperated very clearly, and we can find a linear boundary between these two groups.

The code is as follows.

Problem_4.m

```
clear
close all
clc

%% load data
load kernelpca_data.mat

%% Gaussian Kernel

[D,N] = size(X);

% calculating distance
D1 = L2_distance(X',X')
D2 = D1.^2;

% calculating sigma
[idx,di] = knnsearch(X,X,'k',9);
eighth_idx = idx(:,9);
eighth_nb = di(:,9);
sigma = mean(eighth_nb);

% compute kernel matrix
K = exp(-D2./(2*sigma^2));
J = ones(D,D)/D;

% Centering kernel matrix (non-linearly mapped data).
Kc = K - J*K - K*J + J*K*J;

```

Figure 15: Code part 1

```

% eigendecomposition
[V,S] = eig(Kc);
[dsort, idum]=sort(diag(S),'descend');
l=abs(dsort);
V=V(:,idum);
l2 = l(1:2);

% Two dimensional representation of the data obtained by Kernel PCA
X_proj = V(:,1:2) * diag(sqrt(l2));

f4_1 = figure;plot(X_proj(:,1),X_proj(:,2),'b*','MarkerSize',4);
title 'Gaussian Kernel 2-D'
saveas(f4_1,'/Users/XC/Dropbox/SJSU/Courses/Math 285 Data Modeling/HW/hw2/4_GKernel.jpg','jpg')

%% compare original plot and kernel PCA plot

% original figure
f4_2=figure; plot(X(:,1), X(:,2),'b*','MarkerSize',4)
title 'Scatter plot 1'
saveas(f4_2,'/Users/XC/Dropbox/SJSU/Courses/Math 285 Data Modeling/HW/hw2/4_ORG.jpg','jpg')

```

Figure 16: Code part 2

# 6 Problem 5

(a) (i) Figure 17 shows three true clusters, and Figure 18 shows the results using k-means.
(ii)The error percentage of my clustering is $1 - 0.8933 = 0.1067 = 10.67\%$. I think the error percentage is a little high. So the result is not very good.
(iii)From the following two figures, we can also tell that the boundary of the left two groups are not clear (figure 17), which leads to many misclassification in that small area (figure 18)
(iv) By wikipedia, we find that "A key limitation of k-means is its cluster model. The concept is based on spherical clusters that are separable in a way so that the mean value converges towards the cluster center. " In this case, the shape of each group in our data is ellipse, so this method is not a good one for Iris data.

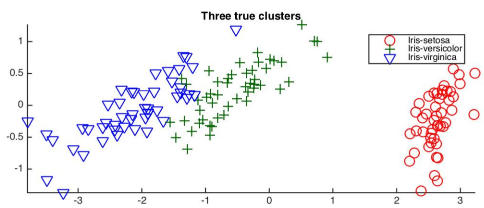


Figure 17: Three true clusters

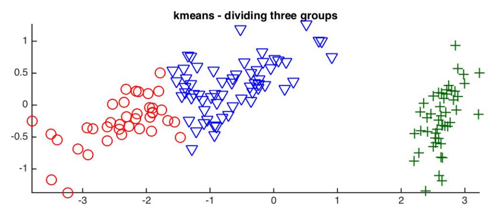


Figure 18: kmeans - dividing three groups

(b) Figure 19 shows the scree plot. From this figure, we could find that k=2 or k=3 are both fine. If we choose k=2, we could find that the two classes ( now Virginica and Versicolor are grouped into one class, setosa is another class) are seperatd clearly. In this case, the misclassification error should be very small.

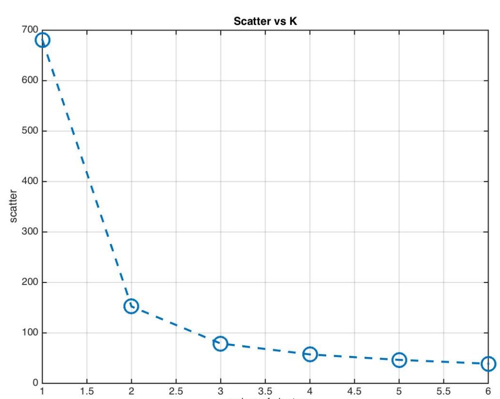


Figure 19: scatter vs k.

The code is as follows.

```
%% Problem 5
clear
close all
clc
%% read data
fileID = fopen('iris.txt');
C = textscan(fileID,'%3.1f, %3.1f, %3.1f, %3.1f, %s');
fclose(fileID);

%% form data matrix and plot data with true labels
X = [C{1:4}]; % concatenate the first four cells to form a matrix

% true lables
labels = zeros(size(X,1),1);
labels(strcmp(C{5}, 'Iris-setosa')) = 1;
labels(strcmp(C{5}, 'Iris-versicolor')) = 2;
labels(strcmp(C{5}, 'Iris-virginica')) = 3;

% display the three true clusters
f5_1 = figure; gcplot(X, labels); axis equal
legend('Iris-setosa','Iris-versicolor','Iris-virginica')
title 'Three true clusters'
saveas(f5_1,'/Users/XC/Dropbox/SJSU/Courses/Math 285 Data Modeling/HW/hw2/5_true.jpg','jpg')

%% perform kmeans
labels_kmeans = kmeans(X, 3, 'Replicates', 10);
f5_2=figure; gcplot(X, labels_kmeans); axis equal
title 'kmeans - dividing three groups'
saveas(f5_2,'/Users/XC/Dropbox/SJSU/Courses/Math 285 Data Modeling/HW/hw2/5_kmeans3.jpg','jpg')

```

Figure 20: Code part 1

```

%% part a: compute clustering accuracy rate
trueLabels = labels;
accuracy = 1 - computing_percentage_of_misclassified_points(labels_kmeans,trueLabels)

%% part b: select number of clusters when not given
k=3
scatter = zeros(1,2*k); % vector of total scatter
scatter(1) = sum(sum((X - repmat(mean(X,1),size(X,1),1)).^2,2)); % j = 1
for j = 2 : 2*k
    [~, ~, intravars] = kmeans(X, j, 'Replicates', 10);
    scatter(j) = sum(intravars);
end

f5_3=figure;
plot(scatter, 'o--', 'markersize', 14, 'linewidth', 2)
xlabel 'number of clusters'
ylabel 'scatter'
title 'Scatter vs K'
grid on
saveas(f5_3,'/Users/XC/Dropbox/SJSU/Courses/Math 285 Data Modeling/HW/hw2/scatter_k.jpg','jpg')

```

Figure 21: Code part 2